



---

**PIR/Mircowave Sensor Flash MCU**

**BA45F6630**

Revision: V1.00 Date: October 08, 2019

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

## Table of Contents

<b>Features</b> .....	<b>6</b>
CPU Features .....	6
Peripheral Features.....	6
<b>General Description</b> .....	<b>7</b>
<b>Block Diagram</b> .....	<b>8</b>
<b>Pin Assignment</b> .....	<b>8</b>
<b>Pin Description</b> .....	<b>9</b>
<b>Absolute Maximum Ratings</b> .....	<b>10</b>
<b>D.C. Electrical Characteristics</b> .....	<b>11</b>
Operating Voltage Characteristics.....	11
Standby Current Characteristics .....	11
Operating Current Characteristics.....	12
<b>A.C. Electrical Characteristics</b> .....	<b>12</b>
High Speed Internal Oscillator – HIRC – Frequency Accuracy .....	12
Low Speed Internal Oscillator Characteristics – LIRC .....	13
Operating Frequency Characteristics Curves .....	13
System Start Up Time Characteristics .....	13
<b>Input/Output Characteristics</b> .....	<b>14</b>
<b>Memory Characteristics</b> .....	<b>14</b>
<b>LVD/LVR Electrical Characteristics</b> .....	<b>15</b>
<b>Reference Voltage Electrical Characteristics</b> .....	<b>15</b>
<b>OP Amplifier Electrical Characteristics</b> .....	<b>15</b>
<b>LDO Electrical Characteristics</b> .....	<b>16</b>
<b>A/D Converter Electrical Characteristics</b> .....	<b>17</b>
<b>Power-on Reset Characteristics</b> .....	<b>17</b>
<b>System Architecture</b> .....	<b>18</b>
Clocking and Pipelining.....	18
Program Counter.....	19
Stack .....	19
Arithmetic and Logic Unit – ALU .....	20
<b>Flash Program Memory</b> .....	<b>20</b>
Structure.....	20
Special Vectors .....	21
Look-up Table.....	21
Table Program Example.....	21
In Circuit Programming – ICP .....	22
On-Chip Debug Support – OCDS .....	23

<b>RAM Data Memory .....</b>	<b>24</b>
Structure.....	24
General Purpose Data Memory .....	24
Special Purpose Data Memory .....	25
<b>Special Function Register Description.....</b>	<b>26</b>
Indirect Addressing Register – IAR0, IAR1 .....	26
Memory Pointers – MP0, MP1 .....	26
Bank Pointer – BP .....	27
Accumulator – ACC.....	27
Program Counter Low Register – PCL .....	27
Look-up Table Registers – TBLP, TBHP, TBLH .....	27
Status Register – STATUS .....	28
<b>EEPROM Data memory .....</b>	<b>29</b>
EEPROM Data Memory Structure .....	29
EEPROM Registers .....	29
Reading Data from the EEPROM .....	31
Writing Data to the EEPROM.....	31
Write Protection.....	31
EEPROM Interrupt.....	31
Programming Considerations.....	32
<b>Oscillators .....</b>	<b>33</b>
Oscillator Overview .....	33
System Clock Configurations .....	33
Internal RC Oscillator – HIRC .....	34
Internal 32kHz Oscillator – LIRC .....	34
Supplementary Oscillator.....	34
<b>Operating Modes and System Clocks .....</b>	<b>35</b>
System Clocks .....	35
System Operation Modes .....	36
Control Register .....	37
Operating Mode Switching .....	39
Standby Current Considerations .....	43
Wake-up .....	43
<b>Watchdog Timer.....</b>	<b>44</b>
Watchdog Timer Clock Source.....	44
Watchdog Timer Control Register .....	44
Watchdog Timer Operation .....	45
<b>Reset and Initialisation.....</b>	<b>46</b>
Reset Functions .....	46
Reset Initial Conditions .....	49
<b>Input/Output Ports .....</b>	<b>51</b>
Pull-high and Pull-low Resistors.....	51
Port A Wake-up .....	52

I/O Port Control Registers .....	52
Pin-shared Functions .....	53
I/O Pin Structures .....	56
Programming Considerations .....	56
<b>Timer Modules – TM .....</b>	<b>57</b>
Introduction .....	57
TM Operation .....	57
TM Clock Source.....	58
TM Interrupts.....	58
TM External Pins.....	58
Programming Considerations.....	59
<b>Standard Type TM – STM .....</b>	<b>60</b>
Standard Type TM Operation.....	60
Standard Type TM Register Description .....	60
Standard Type TM Operation Modes .....	65
<b>Analog to Digital Converter – ADC.....</b>	<b>75</b>
A/D Overview .....	75
A/D Converter Register Description .....	75
A/D Operation .....	78
A/D Reference Voltage.....	79
A/D Converter Input Signal .....	79
Conversion Rate and Timing Diagram .....	79
Summary of A/D Conversion Steps.....	80
Programming Considerations.....	80
A/D Transfer Function .....	80
A/D Programming Examples.....	81
<b>Auto Conversion Function.....</b>	<b>83</b>
Auto Conversion Operation.....	83
Auto Conversion Registers .....	83
<b>Universal Serial Interface Module – USIM .....</b>	<b>85</b>
SPI Interface .....	85
I <sup>2</sup> C Interface .....	93
UART Interface.....	102
<b>LDO Function .....</b>	<b>118</b>
<b>Operational Amplifiers .....</b>	<b>119</b>
Operational Amplifier Registers.....	120
Offset Calibration Procedure.....	122
<b>Interrupts .....</b>	<b>123</b>
Interrupt Registers.....	123
Interrupt Operation.....	127
External Interrupt.....	128
Auto Conversion Function Interrupt .....	128
Timer Module Interrupts .....	129

USIM Interrupt.....	129
Multi-function Interrupt .....	129
A/D Converter Interrupt.....	130
EEPROM Interrupt .....	130
LVD Interrupt .....	130
Time Base Interrupts .....	130
Interrupt Wake-up Function.....	132
Programming Considerations.....	132
<b>Low Voltage Detector – LVD .....</b>	<b>133</b>
LVD Register .....	133
LVD Operation.....	134
<b>Configuration Options.....</b>	<b>135</b>
<b>Application Circuits.....</b>	<b>135</b>
<b>Instruction Set.....</b>	<b>136</b>
Introduction .....	136
Instruction Timing .....	136
Moving and Transferring Data.....	136
Arithmetic Operations.....	136
Logical and Rotate Operation .....	137
Branches and Control Transfer .....	137
Bit Operations .....	137
Table Read Operations .....	137
Other Operations.....	137
<b>Instruction Set Summary .....</b>	<b>138</b>
Table Conventions.....	138
<b>Instruction Definition.....</b>	<b>140</b>
<b>Package Information .....</b>	<b>149</b>
24-pin SSOP (150mil) Outline Dimensions .....	150
SAW Type 24-pin QFN (3mm×3mm×0.55mm) Outline Dimensions .....	151

## Features

### CPU Features

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

### Peripheral Features

- Flash Program Memory: 2K×16
- RAM Data Memory: 256×8
- True EEPROM Memory: 32×8
- Watchdog Timer function
- Up to 15 bidirectional I/O lines
- Dual pin-shared external interrupts
- Two Timer Modules for time measurement, input capture, compare match output, PWM output function or single pulse output function
- Dual Time-Base functions for generation of fixed time interrupt signals
- 4 external channel 12-bit resolution A/D converter
- A/D converter auto conversion function
- A/D converter lower/upper limit preset
- Universal Serial Interfaces Module – USIM for SPI, I<sup>2</sup>C or UART communication
- Dual Operational Amplifiers functions
- LDO function
- Low Voltage Reset function
- Low Voltage Detect function
- Package types: 24-pin QFN, 24-pin SSOP

## General Description

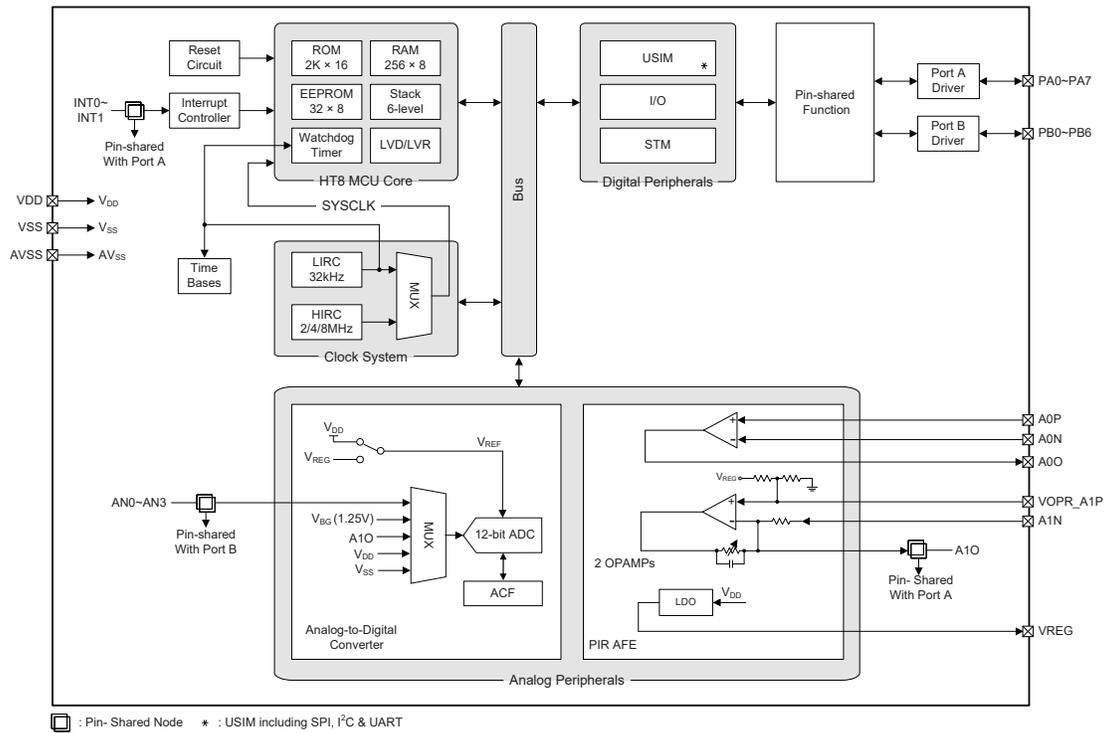
The BA45F6630 is an 8-bit high performance RISC architecture microcontroller, designed especially for PIR/Microwave Sensor applications. Offering users the convenience of Flash Memory multi-programming features, the device also includes wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

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

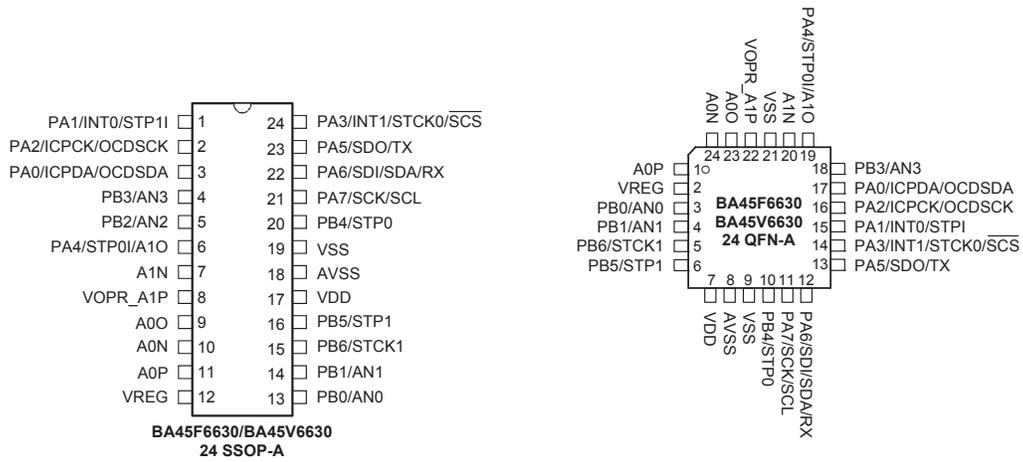
A full choice of internal high and low oscillators functions is provided including two fully integrated system oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives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

The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that only a minimum of external components is required for application implementation, resulting in reduced component costs, low power consumption and high performance. In addition to the above features, the integrated A/D converter and ACF (Auto Conversion Function), cooperated with the external PIR/Microwave Sensor application, provide the device with the versatility for a wide range of PIR/Microwave Sensor products such as PIR bulb, Microwave sensor switch, visitor notification, security monitoring, yard lamp, intelligent appliance to name but a few.

## Block Diagram



## Pin Assignment



- Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.
2. The OCSDA and OCDSCK pins are the OCDS dedicated pins and only available for the BA45V6630 device which is the OCDS EV chip for the BA45F6630 device.
3. For the 24-pin QFN package type, the PB2 pin is an un-bonded pin of which status should be properly configured to avoid the 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.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/ICPDA/ OCDSDA	PA0	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	ICPDA	—	ST	CMOS	ICP address/data pin
	OCDSDA	—	ST	CMOS	OCDS address/data pin, for EV chip only
PA1/INT0/ STP1I	PA1	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT0	INTEG INTC0	ST	—	External interrupt 0
	STP1I	—	ST	—	STM1 capture input
PA2/ICPCK/ OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	ICPCK	—	ST	—	ICP Clock input
	OCDSCK	—	ST	—	OCDS clock input
PA3/INT1/ STCK0/SCS	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT1	INTEG INTC0 PAS0	ST	—	External interrupt 1
	STCK0	PAS0	ST	—	STM0 clock input
	SCS	PAS0	ST	CMOS	SPI slave select pin
PA4/STP0I/A1O	PA4	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	STP0I	PAS0	ST	—	STM0 capture input
	A1O	PAS0	—	AN	OPA1 output
PA5/SDO/TX	PA5	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	SDO	PAS0	—	CMOS	SPI serial data output
	TX	PAS0	—	CMOS	UART TX serial data output
PA6/SDI/SDA/ RX	PA6	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	SDI	PAS0	ST	—	SPI serial data input
	SDA	PAS0	ST	NMOS	I <sup>2</sup> C data line
	RX	PAS0	ST	—	UART RX serial data input
PA7/SCK/SCL	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	SCK	PAS1	ST	CMOS	SPI serial clock
	SCL	PAS1	ST	NMOS	I <sup>2</sup> C clock line
PB0/AN0	PB0	PBPU PBPD PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high or pull-low
	AN0	PBS0	AN	—	A/D Converter analog input

Pin Name	Function	OPT	I/T	O/T	Description
PB1/AN1	PB1	PBPU PBPD PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high or pull-low
	AN1	PBS0	AN	—	A/D Converter analog input
PB2/AN2	PB2	PBPU PBPD PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high or pull-low
	AN2	PBS0	AN	—	A/D Converter analog input
PB3/AN3	PB3	PBPU PBPD PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high or pull-low
	AN3	PBS0	AN	—	A/D Converter analog input
PB4/STP0	PB4	PBPU PBPD PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high or pull-low
	STP0	PBS1	—	CMOS	STM0 output
PB5/STP1	PB5	PBPU PBPD PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high or pull-low
	STP1	PBS1	—	CMOS	STM1 output
PB6/STCK1	PB6	PBPU PBPD	ST	CMOS	General purpose I/O. Register enabled pull-high or pull-low
	STCK1	—	ST	—	STM1 clock input
A0O	—	—	—	AN	OPA0 output
A0P	—	—	AN	—	OPA0 non-inverting input
A0N	—	—	AN	—	OPA0 inverting input
VOPR_A1P	—	—	AN	—	OPA1 non-inverting input
A1N	—	—	AN	—	OPA1 inverting input
VDD	—	—	PWR	—	Positive Power supply
VSS	—	—	PWR	—	Digital Ground
AVSS	—	—	PWR	—	Analog Ground
VREG	—	—	—	PWR	LDO output voltage

Note: I/T: Input type

OPT: Optional by register option

ST: Schmitt Trigger input

NMOS: NMOS output

O/T: Output type

PWR: Power

CMOS: CMOS output

AN: Analog signal

## Absolute Maximum Ratings

Supply Voltage .....	$V_{SS}-0.3V$ to $6.0V$
Input Voltage .....	$V_{SS}-0.3V$ to $V_{DD}+0.3V$
Storage Temperature.....	$-50^{\circ}C$ to $125^{\circ}C$
Operating Temperature.....	$-40^{\circ}C$ to $85^{\circ}C$
$I_{OL}$ 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. Electrical Characteristics

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

### Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V <sub>DD</sub>	Operating Voltage – HIRC	f <sub>sys</sub> =2MHz	2.2	—	5.5	V
		f <sub>sys</sub> =4MHz	2.2	—	5.5	
		f <sub>sys</sub> =8MHz	2.2	—	5.5	
	Operating Voltage – LIRC	f <sub>sys</sub> =32kHz	2.2	—	5.5	

### Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V <sub>DD</sub>	Conditions					
I <sub>STB</sub>	SLEEP Mode	2.2V	WDT off	—	0.08	0.12	1.40	μA
		3V		—	0.08	0.12	1.40	
		5V		—	0.15	0.29	2.20	
		2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3.0	5.0	6.0	
	IDLE0 Mode – LIRC	2.2V	f <sub>SUB</sub> on	—	2.4	4.0	4.8	μA
		3V		—	3.0	5.0	6.0	
		5V		—	5.0	10	12	
	IDLE1 Mode – HIRC	2.2V	f <sub>SUB</sub> on, f <sub>sys</sub> =2MHz	—	80	120	150	μA
		3V		—	110	150	200	
		5V		—	230	350	450	
		2.2V	f <sub>SUB</sub> on, f <sub>sys</sub> =4MHz	—	130	200	250	μA
		3V		—	200	280	350	
		5V		—	210	420	500	
2.2V		f <sub>SUB</sub> on, f <sub>sys</sub> =8MHz	—	300	450	600	μA	
3V			—	400	650	900		
5V			—	700	1000	1200		

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

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

### Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit	
		V <sub>DD</sub>	Conditions					
I <sub>DD</sub>	SLOW Mode – LIRC	2.2V	f <sub>sys</sub> =32kHz	—	8	16	μA	
		3V		—	10	20		
		5V		—	30	50		
	FAST Mode – HIRC	2.2V	f <sub>sys</sub> =2MHz	—	0.15	0.2	mA	
				3V	—	0.2		0.3
				5V	—	0.4		0.6
		2.2V	f <sub>sys</sub> =4MHz	—	0.3	0.5	mA	
				3V	—	0.4		0.6
				5V	—	0.8		1.2
		2.2V	f <sub>sys</sub> =8MHz	—	0.6	1.0	mA	
				3V	—	0.8		1.2
				5V	—	1.6		2.4

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

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

### A.C. Electrical 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>	Temperature				
f <sub>HIRC</sub>	2MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	2	+1%	MHz
			-20°C~60°C	-2%	2	+2%	
			-40°C~85°C	-3%	2	+3%	
		2.2V~5.5V	25°C	-9%	2	+9%	
			-40°C~85°C	-10%	2	+10%	
	4MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	4	+1%	MHz
			-40°C~85°C	-2%	4	+2%	
			-40°C~85°C	-3%	4	+3%	
		2.2V~5.5V	25°C	-2.5%	4	+2.5%	
			-40°C~85°C	-3%	4	+3%	
8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz	
		-40°C~85°C	-10%	8	+2%		
		-40°C~85°C	-10%	8	+3%		
	2.2V~5.5V	25°C	-10%	8	+3%		
		-40°C~85°C	-15%	8	+5%		

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.

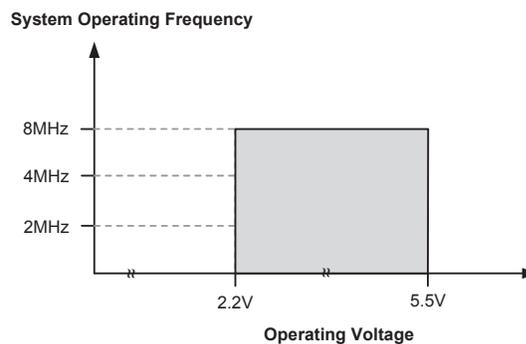
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V<sub>DD</sub> range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

- 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  $\pm 20\%$ .

### Low Speed Internal Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Temperature				
f <sub>LIRC</sub>	LIRC Frequency	3V	25°C Trim@V <sub>DD</sub> =3V	-10%	32	+10%	kHz
		5V	25°C Trim@V <sub>DD</sub> =5V				
		2.2V~5.5V	-40°C~85°C Trim@V <sub>DD</sub> =3V	-60%	32	+60%	
			-40°C~85°C Trim@V <sub>DD</sub> =5V				

### Operating Frequency Characteristics Curves



### System Start Up Time Characteristics

T<sub>a</sub>=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
t <sub>SST</sub>	System Start-up Timer Period of HIRC	—	Power up or wake-up from IDLE/SLEEP mode	—	16	22	t <sub>sys</sub>
	System Start-up Timer Period (With Fast Start-up) of LIRC	—	wake-up from IDLE mode (f <sub>SL</sub> =f <sub>LIRC</sub> )	—	1	4	t <sub>LIRC</sub>
	System Start-up Timer Period (With Fast Start-up) of MCU	—	In IDLE/SLEEP mode, when ADC sent an interrupt signal to MCU, the MCU will fast wake-up in 16 HIRC clocks.	—	—	16	t <sub>sys</sub>
t <sub>RSTD</sub>	System Reset Delay Time (POR Reset)	—	—	25	50	100	ms
	System Reset Delay Time (Any Reset except POR Reset)	—	—	8.3	16.7	33.3	
t <sub>SRESET</sub>	Minimum Software Reset Pulse Width to Reset	—	—	45	90	120	μs

- Note: 1. For the System Start-up time values, whether f<sub>sys</sub> is on or off depends upon the mode type and the chosen f<sub>sys</sub> system oscillator. Details are provided in the System Operating Modes section.
2. The time units, shown by the symbols t<sub>HIRC</sub>, 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. 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	5V	—	0	—	1.5	V
		—	—	0	—	0.2V <sub>DD</sub>	
V <sub>IH</sub>	Input High Voltage for I/O Ports	5V	—	3.5	—	5.0	V
		—	—	0.8V <sub>DD</sub>	—	V <sub>DD</sub>	
I <sub>OL</sub>	Sink Current for I/O Ports	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	6	12	—	mA
		5V		10	25	—	
I <sub>OH</sub>	Source Current for I/O Ports	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-2	-4	—	mA
		5V		-5	-8	—	
R <sub>PH</sub>	Pull-high Resistance for I/O Ports <sup>(1)</sup>	3V	—	40	60	80	kΩ
		5V	—	10	30	50	
R <sub>PL</sub>	Pull-low Resistance for I/O Ports (Port B) <sup>(2)</sup>	3V	—	40	60	80	kΩ
		5V	—	10	30	50	
I <sub>LEAK</sub>	Input Leakage Current	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>	STCKn Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>TPI</sub>	STPnI Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>INT</sub>	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs

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

2. The R<sub>PL</sub> internal pull low resistance value is calculated by connecting to V<sub>DD</sub> and enabling input pin with pull-low resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R<sub>PL</sub> value.

## Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>RW</sub>	V <sub>DD</sub> for Read/Write	—	—	V <sub>DDmin</sub>	—	V <sub>DDmax</sub>	V
<b>Flash Program Memory / Data EEPROM Memory</b>							
t <sub>DEW</sub>	Erase/Write Cycle Time – Flash Program Memory	—	—	—	2	3	ms
	Write Cycle Time – Data EEPROM Memory	—	—	—	4	6	
I <sub>DDPGM</sub>	Programming/Erase Current on V <sub>DD</sub>	—	—	—	—	5.0	mA
E <sub>P</sub>	Cell Endurance – Flash Program Memory	—	—	10K	—	—	E/W
	Cell Endurance – Data EEPROM Memory	—	—	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	—	Device in SLEEP Mode	1.0	—	—	V

## LVD/LVR Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>LVR</sub>	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	-5%	2.10	+5%	V
		—	LVR enable, voltage select 2.55V		2.55		
		—	LVR enable, voltage select 3.15V		3.15		
		—	LVR enable, voltage select 3.8V		3.80		
V <sub>LVD</sub>	Low Voltage Detection Voltage	—	LVD enable, voltage select 2.0V	-5%	2.0	+5%	V
		—	LVD enable, voltage select 2.2V		2.2		
		—	LVD enable, voltage select 2.4V		2.4		
		—	LVD enable, voltage select 2.7V		2.7		
		—	LVD enable, voltage select 3.0V		3.0		
		—	LVD enable, voltage select 3.3V		3.3		
		—	LVD enable, voltage select 3.6V		3.6		
		—	LVD enable, voltage select 4.0V		4.0		
I <sub>LVR</sub>	Additional current for LVR enable	3V	—	—	15	20	μA
		5V		—	20	30	
I <sub>LVD</sub>	Additional current for LVD enable	3V	—	—	15	20	μA
		5V		—	20	30	
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
t <sub>LVDS</sub>	LVDO Stable Time	5V	LVR disable, LVD enable, V <sub>BG</sub> is ready	—	—	100	μs

## Reference Voltage Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>BG</sub>	Bandgap Reference Voltage	—	—	-5%	1.25	+5%	V

Note: The V<sub>BG</sub> voltage is also used as A/D converter signal input.

## OP Amplifier Electrical Characteristics

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>OPA</sub>	Operating Current	5V	OPAnBW[1:0]=00B, no load	—	3.0	5.0	μA
			OPAnBW[1:0]=01B, no load	—	10	16	
			OPAnBW[1:0]=10B, no load	—	80	128	
			OPAnBW[1:0]=11B, no load	—	200	320	
V <sub>OS</sub>	Input Offset Voltage	5V	Without calibration (OnOF[5:0]=100000B)	-15	—	15	mV
			With calibration	-2	—	2	
I <sub>OS</sub>	Input Offset Current	5V	V <sub>IN</sub> = 1/2V <sub>CM</sub>	—	1	10	nA
V <sub>CM</sub>	Common Mode Voltage Range	5V	OPAnBW[1:0]=00B, 01B, 10B, 11B	V <sub>SS</sub>	—	V <sub>DD</sub> -1.4	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
PSRR	Power Supply Rejection Ratio	5V	OPAnBW[1:0]=00B, 01B, 10B, 11B	50	70	—	dB
CMRR	Common Mode Rejection Ratio	5V	OPAnBW[1:0]=00B, 01B, 10B, 11B	50	80	—	dB
A <sub>OL</sub>	Open Loop Gain	5V	OPAnBW[1:0]=00B, 01B, 10B, 11B	60	80	—	dB
SR	Slew Rate	5V	R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPAnBW[1:0]=00B	0.5	1.5	—	V/ms
			R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPAnBW[1:0]=01B	5	15	—	
			R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPAnBW[1:0]=10B	180	500	—	
			R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPAnBW[1:0]=11B	600	1800	—	
GBW	Gain Bandwidth	5V	R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPAnBW[1:0]=00B	2.5	5.0	—	kHz
			R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPAnBW[1:0]=01B	20	40	—	
			R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPAnBW[1:0]=10B	400	600	—	
			R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPAnBW[1:0]=11B	1300	2000	—	
VOR	Maximum Output Voltage Range of A00	5V	OPAnBW[1:0]=00B, 01B R <sub>LOAD</sub> =5kΩ to V <sub>DD</sub> /2	V <sub>SS</sub> +80	—	V <sub>DD</sub> -120	mV
			OPAnBW[1:0]=10B, 11B R <sub>LOAD</sub> =5kΩ to V <sub>DD</sub> /2	V <sub>SS</sub> +40	—	V <sub>DD</sub> -60	
	Maximum Output Voltage Range of A10 which is pin-shared with PA4	5V	OPAnBW[1:0]=00B, 01B R <sub>LOAD</sub> =5kΩ to V <sub>DD</sub> /2	V <sub>SS</sub> +140	—	V <sub>DD</sub> -160	mV
			OPAnBW[1:0]=10B, 11B R <sub>LOAD</sub> =5kΩ to V <sub>DD</sub> /2	V <sub>SS</sub> +120	—	V <sub>DD</sub> -140	
I <sub>SC</sub>	Output Short Circuit Current	5V	R <sub>LOAD</sub> =5.1Ω, OPAnBW[1:0]=00B, 01B	±6	±12	—	mA
			R <sub>LOAD</sub> =5.1Ω, OPAnBW[1:0]=10B, 11B	±10	±20	—	

## LDO Electrical Characteristics

$$V_{IN}=V_{OUT}+1V, T_a=-40^{\circ}C-85^{\circ}C$$

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>IN</sub>	Supply Voltage	—	—	2.7	3.3	5.5	V
V <sub>OUT</sub>	Output Voltage	—	V <sub>REG</sub> output decided by VSEL fields	-3%	V <sub>REG</sub>	3%	V
I <sub>OUT</sub>	Driving Current	5V	V <sub>IN</sub> =5V, VCAP=0.1μF	1	—	—	mA
I <sub>Q</sub>	Quiescent Current	—	After startup, no load, include bandgap consumption	3	—	8	μA

Note: 1. This LDO can provide stable power supply for PIR sensor with a 10μF capacitor.

2. The VREG pin should be connected to 0.1μF capacitor for ADC reference voltage and 10μF capacitor for PIR sensor.

## A/D Converter Electrical Characteristics

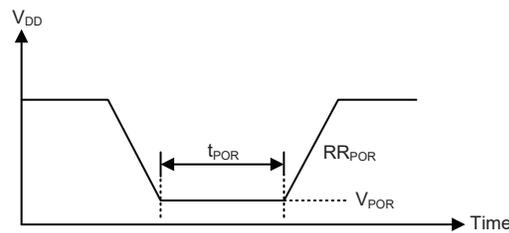
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	A/D Converter Operating Voltage	—	—	2.7	—	5.5	V
V <sub>ADI</sub>	A/D Converter Input Voltage	—	—	0	—	V <sub>DD</sub> / V <sub>REF</sub>	V
DNL	Differential Non-linearity	2.7V	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs	-3	—	+3	LSB
		3V					
		5V					
INL	Integral Non-linearity	2.7V	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs	-4	—	+4	LSB
		3V					
		5V					
I <sub>ADC</sub>	Additional Current for ADC Enable	3V	ADM=0	—	0.6	1.3	mA
			ADM=1	—	0.7	1.5	
		5V	ADM=0	—	1.0	2.0	
t <sub>ADCK</sub>	A/D Converter Clock Period	2.7V~5.5V	—	0.5	—	10	μs
t <sub>ADC</sub>	A/D Conversion Time (Include Sample and Hold Time)	2.7V~5.5V	—	16	—	20	t <sub>ADCK</sub>
t <sub>ADS</sub>	A/D Converter Sampling Time	2.7V~5.5V	—	—	4	—	t <sub>ADCK</sub>
t <sub>ON2ST</sub>	A/D Converter On-to-Start Time	2.7V~5.5V	—	4	—	—	μs

## Power-on Reset Characteristics

Ta=25°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



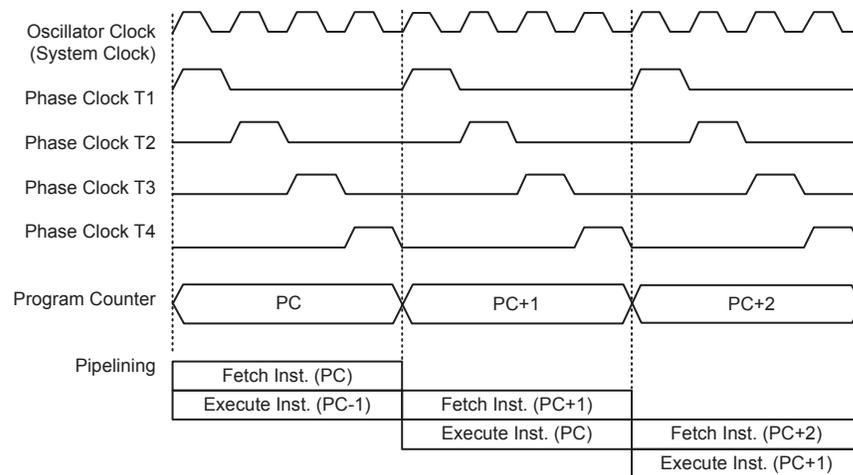
## System Architecture

A key factor in the high-performance features of the 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 cycle, with the exception of branch or call instructions which need one more cycle. An 8-bit wide ALU is used in practically all instruction set operations, which carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O and A/D control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

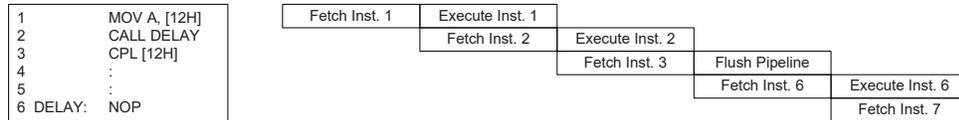
### Clocking and Pipelining

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

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



**System Clocking and Pipelining**



**Instruction Fetching**

### Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as “JMP” or “CALL” that 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	
Program Counter High Byte	PCL Register
PC10~PC8	PCL7~PCL0

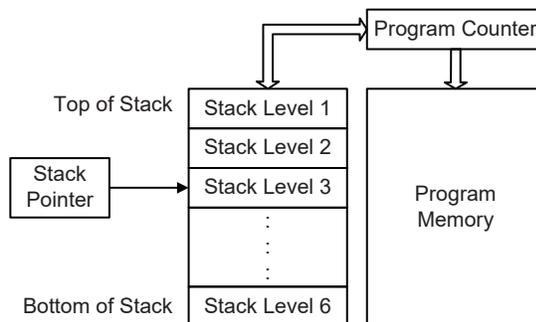
**Program Counter**

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

### Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 6 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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



### Arithmetic and Logic Unit – ALU

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

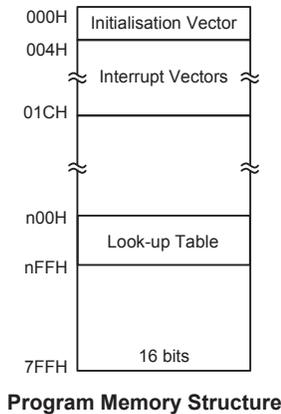
- Arithmetic operations:  
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations:  
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation:  
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement:  
INCA, INC, DECA, DEC
- Branch decision:  
JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

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



### Special Vectors

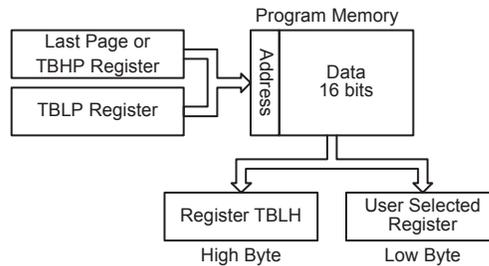
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 0000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

### 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 “TABRD [m]” or “TABRDL [m]” instructions, respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as 0.

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



### Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is “0700H” which refers to the start address of the last page within the 2K words Program Memory of the 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 “0706H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specified

address pointed by the TBLP and TBHP register if the “TABRD [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]” instruction is executed.

Because the TBLH register is a read-only register and cannot 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, 07h         ; initialise high table pointer
mov tbhp, a
:
tabrd tempreg1     ; transfers value in table referenced by table pointer data at program
                  ; memory address "0706H" transferred to tempreg1 and TBLH
dec tblp           ; reduce value of table pointer by one
tabrd tempreg2     ; transfers value in table referenced by table pointer data at program
                  ; memory address "0705H" transferred to tempreg2 and TBLH in this
                  ; example the data 1AH is transferred to tempreg1 and data "0FH" to
                  ; register tempreg2
                  ; the value "00H" will be transferred to the high byte register TBLH
:
org 0700h         ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:

```

**In Circuit Programming – ICP**

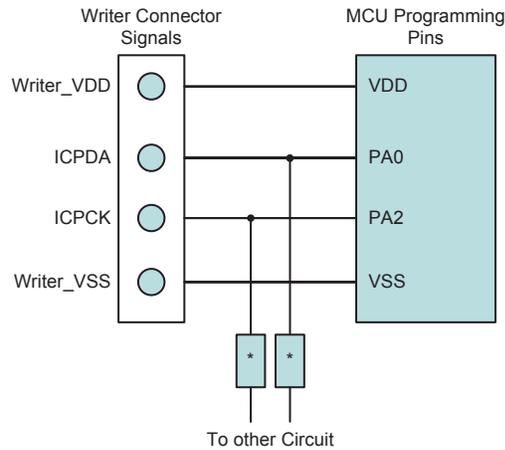
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides 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 Holtek Flash MCU to Writer Programming Pin correspondence table is as follows:

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VDD	VDD	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, taking control of the ICPDA and ICPCCK pins for data and clock programming purposes. The user must there take care 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

An EV chip exists for the purposes of device emulation. This EV chip device also provides an “On-Chip Debug” function to debug the device during the development process. The EV chip and the actual MCU device are almost functionally compatible except for the “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 actual MCU device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For a more detailed OCDS description, 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

## RAM 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.

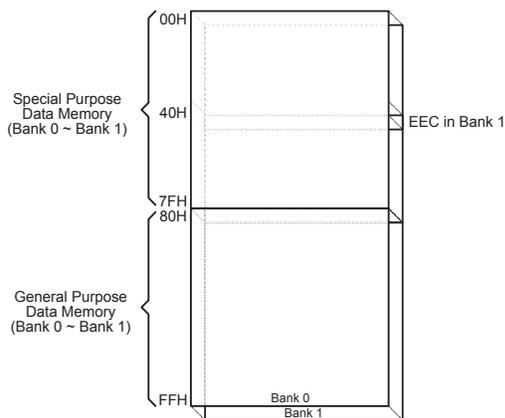
### Structure

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

The overall Data Memory is subdivided into two banks. The Special Purpose Data Memory registers are accessible in all banks, with the exception of the EEC register at address 40H, which is only accessible in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory for the device is the address 00H.

Special Purpose Data Memory	General Purpose Data Memory	
Located Banks	Capacity	Bank: Address
0, 1	256×8	0: 80H~FFH 1: 80H~FFH

**Data Memory Summary**



**Data Memory Structure**

### General Purpose Data Memory

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

### Special Purpose Data Memory

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

	Bank 0	Bank 1		Bank 0	Bank 1
00H	IAR0	IAR0	40H		EEC
01H	MP0	MP0	41H	STM1C0	STM1C0
02H	IAR1	IAR1	42H	STM1C1	STM1C1
03H	MP1	MP1	43H	STM1DL	STM1DL
04H	BP	BP	44H	STM1DH	STM1DH
05H	ACC	ACC	45H	STM1AL	STM1AL
06H	PCL	PCL	46H	STM1AH	STM1AH
07H	TBLP	TBLP	47H	SIMC0	SIMC0
08H	TBLH	TBLH	48H	SIMC1/UUCR1	SIMC1/UUCR1
09H	TBHP	TBHP	49H	SIMC2/SIMA/UUCR2	SIMC2/SIMA/UUCR2
0AH	STATUS	STATUS	4AH	SIMD/UTXR_RXR	SIMD/UTXR_RXR
0BH	INTC0	INTC0	4BH	SIMTOC/UBRG	SIMTOC/UBRG
0CH	INTC1	INTC1	4CH	UUSR	UUSR
0DH	TB0C	TB0C	4DH		
0EH	TB1C	TB1C	4EH		
0FH	PSCR	PSCR	4FH		
10H			50H		
11H			51H		
12H	PA	PA	52H		
13H	PAC	PAC	53H		
14H	PAPU	PAPU	54H		
15H	PAWU	PAWU	55H		
16H			56H		
17H	OPAC2	OPAC2	57H		
18H	LVDC	LVDC	58H		
19H	LVRC	LVRC	59H		
1AH	LDOC	LDOC	5AH		
1BH	OPA0VOS	OPA0VOS	5BH		
1CH	OPA1VOS	OPA1VOS	5CH		
1DH	WDTC	WDTC	5DH		
1EH	STM0C0	STM0C0	5EH		
1FH	STM0C1	STM0C1	5FH		
20H	STM0DL	STM0DL	60H		
21H	STM0DH	STM0DH	61H		
22H	STM0AL	STM0AL	62H		
23H	STM0AH	STM0AH	63H		
24H	ADRL	ADRL	64H		
25H	ADRH	ADRH	65H		
26H	ADCR0	ADCR0	66H		
27H	ADCR1	ADCR1	67H		
28H	PAS0	PAS0	68H		
29H	PAS1	PAS1	69H		
2AH	PBS0	PBS0	6AH		
2BH	PBS1	PBS1	6BH		
2CH	SMOD1	SMOD1	6CH		
2DH	SMOD	SMOD	6DH		
2EH	INTEG	INTEG	6EH		
2FH	OPAC0	OPAC0	6FH		
30H	OPAC1	OPAC1	70H		
31H	ACFC0	ACFC0	71H		
32H	ACFC1	ACFC1	72H		
33H	MF10C	MF10C	73H		
34H	MF11C	MF11C	74H		
35H	PB	PB	75H		
36H	PBC	PBC	76H		
37H	PBPU	PBPU	77H		
38H	PBPD	PBPD	78H		
39H			79H		
3AH	EEA	EEA	7AH		
3BH	EED	EED	7BH		
3CH	LULV	LULV	7CH		
3DH	HULV	HULV	7DH		
3EH	LLLV	LLLV	7EH		
3FH	HLLV	HLLV	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 Register – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, 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 and IAR1 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 or MP1. Acting as a pair, IAR0 and MP0 can together access data from Bank 0 while the IAR1 and MP1 register pair can access data from any bank. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of “00H” and writing to the registers indirectly will result in no operation.

### Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 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 Bank 0, while MP1 and IAR1 are used to access data from all banks according to BP register. Direct Addressing can only be used with Bank 0, all other Banks must be addressed indirectly using MP1 and IAR1.

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

```
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:
```

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

### Bank Pointer – BP

For this device, the Data Memory is divided into two banks, Bank 0 and Bank 1. Selecting the required Data Memory area is achieved using the Bank Pointer. Bit 0 of the Bank Pointer is used to select Data Memory Banks 0~1.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the IDLE/SLEEP Mode, in which case, the Data Memory bank remains unaffected. It should be noted that the Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within any bank. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from Bank 1 must be implemented using Indirect Addressing.

#### • BP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	DMBP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as “0”

Bit 0 **DMBP0**: Select Data Memory Banks  
 0: Bank 0  
 1: Bank 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 Register – PCL

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

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

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

## Status Register – STATUS

This 8-bit register contains the 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 and C 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.

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	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R	R	R/W	R/W	R/W	R/W
POR	—	—	0	0	x	x	x	x

“x”: Unknown

- Bit 7~6      Unimplemented, read as “0”
- Bit 5        **TO**: Watchdog Time-Out flag  
0: After power up or executing the “CLR WDT” or “HALT” instruction  
1: A watchdog time-out occurred
- Bit 4        **PDF**: Power down flag  
0: After power up or executing the “CLR WDT” instruction  
1: By executing the “HALT” instruction
- Bit 3        **OV**: Overflow flag  
0: No overflow  
1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa.
- Bit 2        **Z**: Zero flag  
0: The result of an arithmetic or logical operation is not zero  
1: The result of an arithmetic or logical operation is zero

- Bit 1     **AC:** Auxiliary flag  
           0: No auxiliary carry  
           1: An operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction
- Bit 0     **C:** Carry flag  
           0: No carry-out  
           1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation
- The C flag is also affected by a rotate through carry instruction.

## EEPROM Data memory

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

### EEPROM Data Memory Structure

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

### EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Bank 1, cannot be directly addressed and can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

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

EEPROM Register List

#### • EEA Register

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

- Bit 7~5     Unimplemented, read as “0”
- Bit 4~0     **EEA4~EEA0:** Data EEPROM address  
               Data EEPROM address bit 4 ~ bit 0

• **EED Register**

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

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

• **EEC Register**

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

Bit 7~4     Unimplemented, read as “0”

Bit 3     **WREN**: Data EEPROM Write Enable  
0: Disable  
1: Enable

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

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

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

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

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

Bit 0     **RD**: EEPROM Read Control  
0: Read cycle has finished  
1: Activate a read cycle

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

- Note: 1. The WREN, WR, RDEN and RD cannot be set high at the same time in one instruction. The WR and RD cannot be set high at the same time.  
2. Ensure that the  $f_{SUB}$  clock is stable before executing the write operation.  
3. Ensure that the write operation is totally complete before changing the EEC register content.

## Reading Data from the EEPROM

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

## Writing Data to the EEPROM

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

## Write Protection

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

## EEPROM Interrupt

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

## Programming Considerations

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

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

## Programming Examples

### Reading Data from the EEPROM – Polling Method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, 040H              ; setup memory pointer MP1
MOV MP1, A               ; MP1 points to EEC register
MOV A, 01H               ; setup Bank Pointer
MOV BP, A
SET IAR1.1               ; set RDEN bit, enable read operations
SET IAR1.0               ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                ; check for read cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read if no more read operations are required
CLR BP
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

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

### Writing Data to the EEPROM – Polling Method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA       ; user defined data
MOV EED, A
MOV A, 040H              ; setup memory pointer MP1
MOV MP1, A               ; MP1 points to EEC register
MOV A, 01H               ; setup Bank Pointer
MOV BP, A
CLR EMI
SET IAR1.3               ; set WREN bit, enable write operations
SET IAR1.2               ; start Write Cycle - set WR bit - executed immediately
                        ; after set WREN bit

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

## Oscillators

Various oscillator options offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving. Oscillator selections and operation are selected through 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. Fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options and operations are selected through a combination of configuration options and relevant control registers. The higher frequency oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

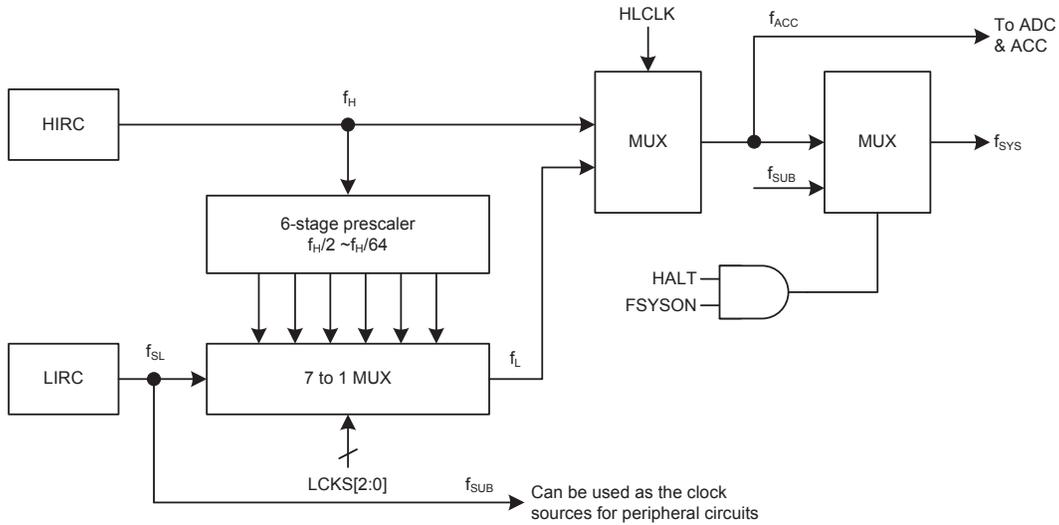
Type	Name	Frequency
Internal High Speed RC	HIRC	2, 4, 8MHz
Internal Low Speed RC	LIRC	32kHz

**Oscillator Types**

### System Clock Configurations

There are two methods of generating the system clock, one high speed oscillator and one low speed oscillator. The high speed oscillator is the internal 2/4/8MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and LCKS2~LCKS0 bits in the SMOD register and as the system clock can be dynamically selected.

The actual source clock used for the high speed oscillator is chosen via configuration options. The frequency of the slow speed or high speed system clock is also determined using the HLCLK bit and LCKS2~LCKS0 bits in the SMOD register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillator. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.



**System Clock Configurations**

**Internal RC Oscillator – HIRC**

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 2MHz, 4MHz or 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

**Internal 32kHz Oscillator – LIRC**

The Internal 32kHz System Oscillator is the low frequency oscillator. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation.

**Supplementary Oscillator**

The low speed oscillator, in addition to providing a system clock source is also used to provide a clock source to other device functions.

## Operating Modes and System Clocks

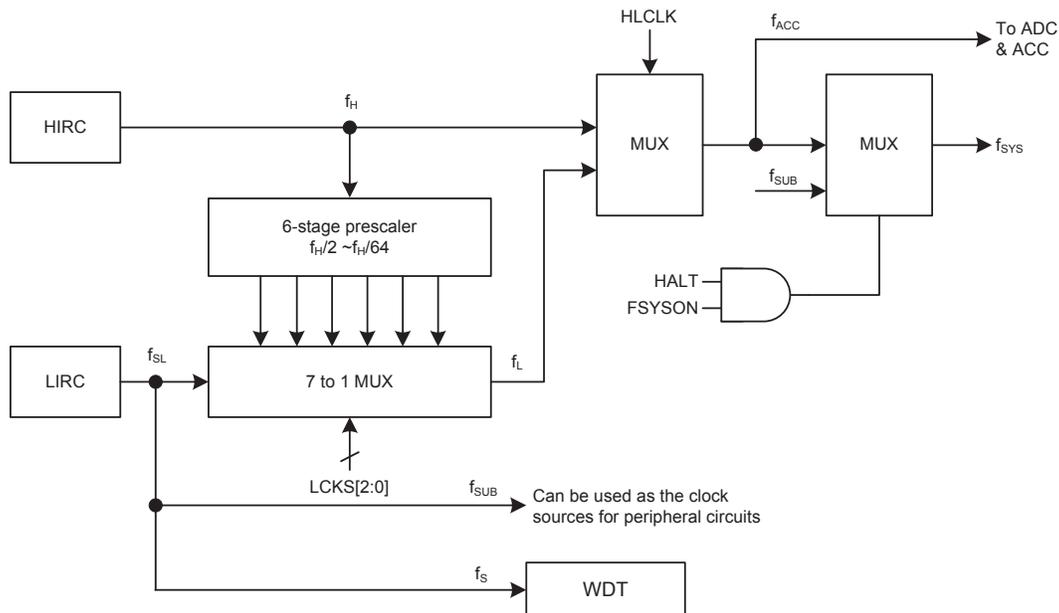
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their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

### System Clocks

The device has two different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock options using configuration options and 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  $f_L$  source, and is selected using the HLCLK bit and LCKS2~LCKS0 bits in the SMOD register. The high speed system clock can be sourced from HIRC oscillator. The low speed system clock source can be sourced from internal clock  $f_{SL}$  which is sourced by LIRC oscillator or a divided version of the high speed system oscillator has a range of  $f_H/2 \sim f_H/64$ .

There are two additional internal clocks for the peripheral circuits, the substitute clock,  $f_{SUB}$ , and  $f_S$ . Each of these internal clocks is sourced by the LIRC oscillator. The  $f_{SUB}$  clock is used as the clock sources for peripheral circuit.



Note: When the system clock source  $f_{SYS}$  is switched to  $f_L$  from  $f_H$  and  $f_L$  comes from  $f_{SL}$ , the  $f_H$  will stop to conserve the power. Thus there is no  $f_H \sim f_H/64$  for peripheral circuit to use.

### Device Clock Configurations

## 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 SLEEP0, SLEEP1, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.

Operation Mode	Description				
	CPU	f <sub>sys</sub>	f <sub>sub</sub>	f <sub>s</sub>	f <sub>acc</sub> (Note)
FAST Mode	ON	f <sub>H</sub>	ON	ON	OFF
SLOW Mode	ON	f <sub>SL</sub> or f <sub>H</sub> /2~f <sub>H</sub> /64	ON	ON	OFF
IDLE0 Mode	OFF	OFF	ON	ON	ON
IDLE1 Mode	OFF	ON	ON	ON	ON
SLEEP0 Mode	OFF	OFF	OFF	OFF	OFF
SLEEP1 Mode	OFF	OFF	ON	ON	ON

Note: f<sub>acc</sub> is made for PIR application. When CPU enters HALT state, this clock is supplied to ADC and ACC circuit.

### FAST Mode

This is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source will come from the HIRC oscillator.

### 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 the LIRC oscillator or a divided version of the high speed system oscillator. The high speed oscillator will however first be divided by a ratio ranging from 2 to 64, the actual ratio being selected by the LCKS2~LCKS0 and HLCLK bits in the SMOD register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

### SLEEP0 Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP0 mode the CPU will be stopped, and the f<sub>sub</sub>, f<sub>s</sub> and f<sub>acc</sub> clocks will be stopped too, and the Watchdog Timer function is disabled. In this mode, the LVDEN is must set to "0". If the LVDEN is set to "1", it won't enter the SLEEP0 Mode.

### SLEEP1 Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP1 mode the CPU will be stopped. However the f<sub>sub</sub>, f<sub>s</sub> and f<sub>acc</sub> clocks will continue to operate if the LVDEN is "1" or the Watchdog Timer function is enabled.

### IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYS0N bit in the SMOD1 register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU but some peripheral functions will remain operational such as the Watchdog Timer. In the IDLE0 Mode, the system oscillator will be stopped.

**IDLE1 Mode**

The IDLE1 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the SMOD1 register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU but may continue to provide a clock source to keep some peripheral functions operational such as the Watchdog Timer. In the IDLE1 Mode, the system oscillator will continue to run, and this system oscillator may be high speed or low speed system oscillator.

**Control Register**

The registers, SMOD and SMOD1, are used to control the system clock and the corresponding oscillator configurations.

• **SMOD Register**

Bit	7	6	5	4	3	2	1	0
Name	LCKS2	LCKS1	LCKS0	—	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	—	R	R	R/W	R/W
POR	0	0	0	—	0	0	1	1

Bit 7~5 **LCKS2~LCKS0**: The low frequency system clock selection when HLCLK is “0”

- 000:  $f_L = f_{SL} (f_{LIRC})$
- 001:  $f_L = f_{SL} (f_{LIRC})$
- 010:  $f_L = f_H / 64$
- 011:  $f_L = f_H / 32$
- 100:  $f_L = f_H / 16$
- 101:  $f_L = f_H / 8$
- 110:  $f_L = f_H / 4$
- 111:  $f_L = f_H / 2$

These three bits are used to select which clock is used as the low frequency system clock source. In addition to the system clock source, which is the LIRC, 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 **LTO**: Low speed system oscillator ready flag

- 0: Not ready
- 1: Ready

This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP0 Mode but after a wake-up has occurred, the flag will change to a high level after 1~2 clock cycles if the LIRC oscillator is used.

Bit 2 **HTO**: High speed system oscillator ready flag

- 0: Not ready
- 1: Ready

This is the high speed system oscillator ready flag which indicates when the high speed system oscillator is stable. This flag is cleared to “0” by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable.

Therefore this flag will always be read as “0” by the application program after device power-on. The flag will be low when in the SLEEP or IDLE0 Mode but after a wake-up has occurred, the flag will change to a high level after 15~16 clock cycles if the HIRC oscillator is used.

- Bit 1      **IDLEN:** IDLE Mode control  
             0: Disable  
             1: Enable  
             This is the IDLE Mode Control bit and determines what happens when the HALT instruction is executed. If this bit is high, when a HALT instruction is executed the device will enter the IDLE Mode. In the IDLE1 Mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if FSYSON bit is high. If FSYSON bit is low, the CPU and the system clock will all stop in IDLE0 mode. If the bit is low the device will enter the SLEEP Mode when a HALT instruction is executed.
- Bit 0      **HLCLK:** system clock selection  
             0:  $f_H/2 \sim f_H/64$  or  $f_{SL}$   
             1:  $f_H$

• **SMOD1 Register**

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

“x”: Unknown

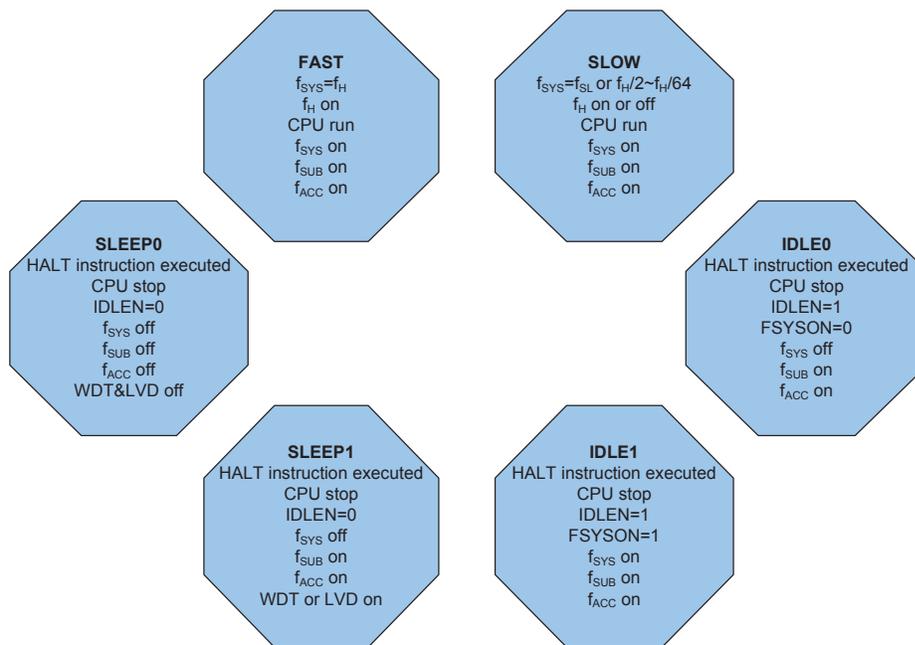
- Bit 7      **FSYSON:**  $f_{SYS}$  Control in IDLE Mode  
             0: Disable  
             1: Enable
- Bit 6~3    Unimplemented, read as “0”
- Bit 2      **LVRF:** LVR function reset flag  
             Refer to the “Low Voltage Reset” section
- Bit 1      **LRF:** LVR Control register software reset flag  
             Refer to the “Low Voltage Reset” section
- Bit 0      **WRF:** WDT Control register software reset flag  
             Refer to the “Watchdog Timer” section

## 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 HLCLK bit and LCKS2~LCKS0 bits in the SMOD 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 IDLEN bit in the SMOD register and FSYSON in the SMOD1 register.

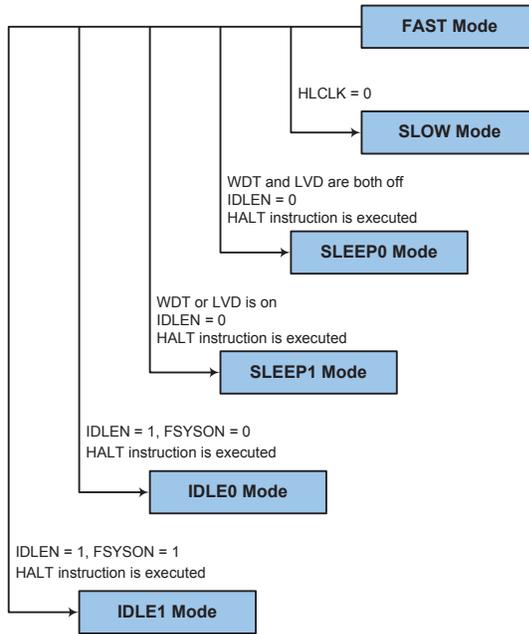
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source,  $f_H$ , to the clock source,  $f_H/2 \sim f_H/64$  or  $f_{SL}$ . If the clock is from the  $f_{SL}$ , the high speed clock source will stop running to conserve power. When this happens it must be noted that the  $f_H/16$  and  $f_H/64$  internal clock sources will also stop running, which may affect the operation of other internal functions such as the Timers. The accompanying flowchart shows what happens when the device moves between the various operating modes.



**FAST Mode to SLOW Mode Switching**

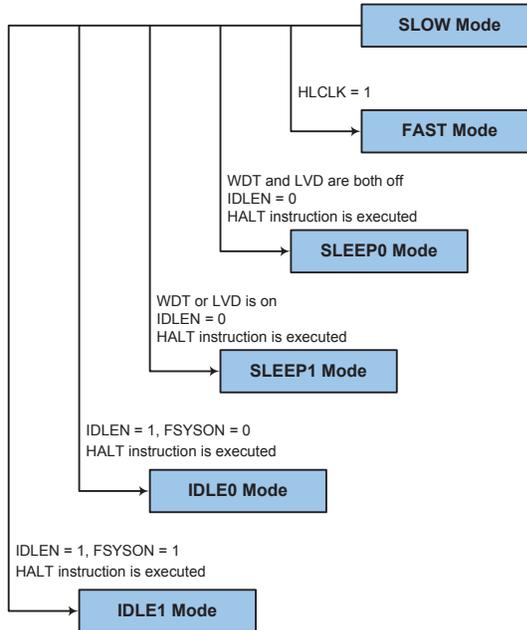
When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to “0”. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

The SLOW Mode system clock is sourced from the LIRC oscillator or a divided version of the HIRC oscillator and therefore requires these oscillators to be stable before full mode switching occurs. This is monitored using the LTO bit or HTO bit in the SMOD register.



### SLOW Mode to FAST Mode Switching

In SLOW Mode the system uses the LIRC low speed system oscillator or the HIRC high speed system oscillator. To switch back to the FAST Mode, the HLCLK bit should be set to “1”. As a certain amount of time will be required for the high frequency clock to stabilise, the status of the HTO bit is checked.



### Entering the SLEEP0 Mode

There is only one way for the device to enter the SLEEP0 Mode and that is to execute the “HALT” instruction in the application program with the IDLEN bit in SMOD register equal to “0” and the WDT and LVD both off. When this instruction is executed under the conditions described above, the following will occur:

- The system clock, WDT 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 WDT will be cleared and stopped.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

### Entering the SLEEP1 Mode

There is only one way for the device to enter the SLEEP1 Mode and that is to execute the “HALT” instruction in the application program with the IDLEN bit in SMOD register equal to “0” and the WDT or LVD on. 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, but the WDT or LVD will remain with the clock source coming from the  $f_{SL}$  clock.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT is enabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

### 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 IDLEN bit in SMOD register equal to “1” and the FSYSON bit in SMOD1 register equal to “0”. 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, but the  $f_{SUB}$  clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT is enabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

### 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 the IDLEN bit in SMOD register equal to “1” and the FSYSON bit in SMOD1 register equal to “1”. When this instruction is executed under the conditions described above, the following will occur:

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

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

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs.

In the IDLE1 Mode the system oscillator is on, if the system oscillator is from the high speed system oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

## Wake-up

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

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

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

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a 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,  $f_s$ , which is in turn supplied by the LIRC oscillator. The LIRC internal oscillator has an approximate period of 32kHz at a supply voltage of 5V. However, it should be noted that 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

If the WDT configuration option is selected as “Always Enable”:

10101 or 01010: Enable

Other: Reset MCU

If the WDT configuration option is selected as “By WDTC Control”:

10101: Disable

01010: Enable

Other Values: 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 SMOD1 register will be set to 1.

Bit 2~0

**WS2~WS0:** WDT time-out period selection

000:  $2^8/f_s$

001:  $2^{10}/f_s$

010:  $2^{12}/f_s$

011:  $2^{14}/f_s$

100:  $2^{15}/f_s$

101:  $2^{16}/f_s$

110:  $2^{17}/f_s$

111:  $2^{18}/f_s$

• **SMOD1 Register**

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

“x”: Unknown

- Bit 7      **FSYSON**:  $f_{SYS}$  Control in IDLE Mode  
Refer to the “Operating Modes and System Clocks” section
- Bit 6~3    Unimplemented, read as “0”
- Bit 2      **LVRF**: LVR function reset flag  
Refer to the “Low Voltage Reset” section
- Bit 1      **LRF**: LVR Control register software reset flag  
Refer to the “Low Voltage Reset” section
- Bit 0      **WRF**: WDT Control register software reset flag  
0: Not occur  
1: Occurred  
This bit is set to 1 by the WDT Control register software reset and cleared to 0 by the application program. Note that this bit can only be cleared to 0 by the application program.

**Watchdog Timer Operation**

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer additional enable/disable and reset control of the Watchdog Timer. If the WDT configuration option has selected that the WDT function is always enabled, then WE4~WE0 bits still have effect on the WDT function. When the WE4~WE0 bits value are equal to 01010B or 10101B, the WDT function is enabled. However, if the WE4~WE0 bits are changed to any other values except 01010B and 10101B, which could be caused by adverse environmental conditions such as noise, it will reset the device after a delay time,  $t_{SRESET}$ . If the WDT configuration option has selected that the WDT function is controlled by the WDTC register, then the WDT control register bits, WE4~WE0, are used to enable or disable the Watchdog Timer. In this case the WDT function will be disabled when the WE4~WE0 bits are equal to 10101B and 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.

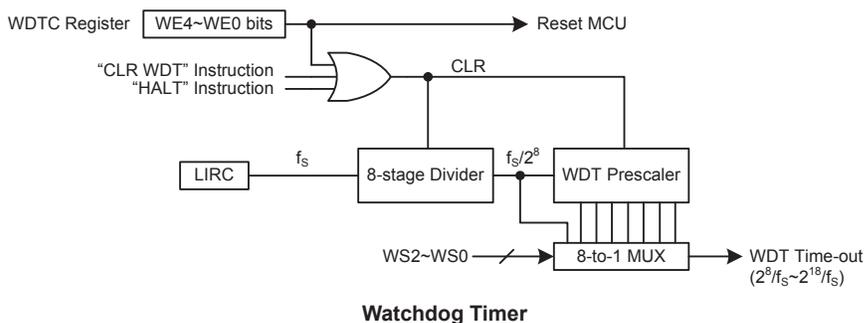
WDT Configuration Options	WE4~WE0 Bits	WDT Function
Always Enable	01010B or 10101B	Enable
	Any other value	Reset MCU
By WDTC Control	10101B	Disable
	01010B	Enable
	Any other value	Reset MCU

**Watchdog Timer Function Control**

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

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

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.

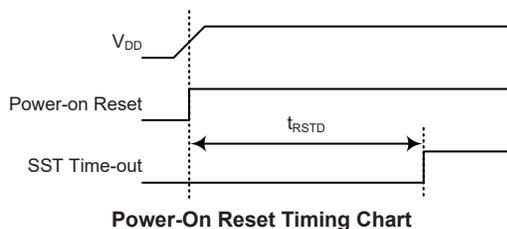
Another type of reset is when the Watchdog Timer overflows and resets. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, is implemented in situations where the power supply voltage falls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

### Reset Functions

There are several ways in which a reset can occur, each of which will be described as follows.

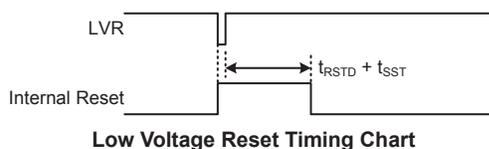
#### Power-on Reset

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.



**Low Voltage Reset – LVR**

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled in FAST and SLOW mode with a specific LVR voltage  $V_{LVR}$ . 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 SMOD1 register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between  $0.9V \sim V_{LVR}$  must exist for a time greater than that specified by  $t_{LVR}$  in the 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 LVS 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 SMOD1 register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the 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								
POR	0	1	0	1	0	1	0	1

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

- 01010101: 2.1V
- 00110011: 2.55V
- 10011001: 3.15V
- 10101010: 3.8V

Any other value: Generates MCU reset – Register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the four 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 the four defined LVR values above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time,  $t_{SRESET}$ . However in this situation the register contents will be reset to the POR value.

• **SMOD1 Register**

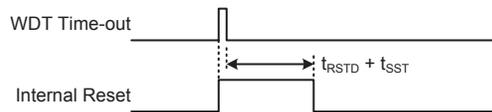
Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	—	LVRF	LRF	WRF
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	x	0	0

“x”: Unknown

- Bit 7      **FSYSON**:  $f_{SYS}$  Control in IDLE Mode  
Refer to the “Operating Modes and System Clocks” section
- Bit 6~3    Unimplemented, read as “0”
- Bit 2      **LVRF**: LVR function reset flag  
0: Not occur  
1: Occurred  
This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.
- Bit 1      **LRF**: LVR Control register software reset flag  
0: Not occur  
1: Occurred  
This bit is set to 1 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 0 by the application program.
- Bit 0      **WRF**: WDT Control register software reset flag  
Refer to the “Watchdog Timer” section

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

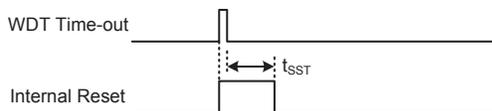
The Watchdog time-out Reset during normal operation is the same as the hardware LVR reset except that the Watchdog time-out flag TO will be set to “1”.



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

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

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



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

### Reset Initial Conditions

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

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

"u": Unchanged

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

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

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

Register	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
MP0	x xxx x xxx	x xxx x xxx	x xxx x xxx	u u u u u u u u
MP1	x xxx x xxx	x xxx x xxx	x xxx x xxx	u u u u u u u u
BP	- - - - - 0	- - - - - 0	- - - - - 0	- - - - - u
ACC	x xxx x xxx	u u u u u u u u	u u u u u u u u	u u u u u u u u
PCL	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
TBLP	x xxx x xxx	u u u u u u u u	u u u u u u u u	u u u u u u u u
TBLH	x xxx x xxx	u u u u u u u u	u u u u u u u u	u u u u u u u u
TBHP	- - - - - x x x	- - - - - u u u	- - - - - u u u	- - - - - u u u
STATUS	- - 0 0 x xxx	- - u u u u u u	- - 1 u u u u u	- - 1 1 u u u u
INTC0	- 0 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0 0	- u u u u u u u
INTC1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
TB0C	0 - - - - 0 0 0 0	0 - - - - 0 0 0 0	0 - - - - 0 0 0 0	u - - - - u u u u
TB1C	0 - - - - 0 0 0 0	0 - - - - 0 0 0 0	0 - - - - 0 0 0 0	u - - - - u u u u
PSCR	- - - - - - 0 0	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
PA	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PAC	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PAPU	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PAWU	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
OPAC2	0 - 0 0 0 - 0 0	0 - 0 0 0 - 0 0	0 - 0 0 0 - 0 0	u - u u u - u u
LVDC	- - 0 0 0 0 0 0	- - 0 0 0 0 0 0	- - 0 0 0 0 0 0	- - u u u u u u
LVRC	0 1 0 1 0 1 0 1	u u u u u u u u	0 1 0 1 0 1 0 1	u u u u u u u u

Register	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
LDOC	0000 0000	0000 0000	0000 0000	uuuu uuuu
OPA0VOS	0010 0000	0010 0000	0010 0000	uuuu uuuu
OPA1VOS	0010 0000	0010 0000	0010 0000	uuuu uuuu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
STM0C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0DH	---- --00	---- --00	---- --00	---- --uu
STM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0AH	---- --00	---- --00	---- --00	---- --uu
ADRL	0000 ----	0000 ----	0000 ----	uuuu ----
ADRH	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADCR0	0110 0000	0110 0000	0110 0000	uuuu uuuu
ADCR1	1--0 0000	1--0 0000	1--0 0000	u--u uuuu
PAS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	---- --00	---- --00	---- --00	---- --uu
PBS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS1	---- 0000	---- 0000	---- 0000	---- uuuu
SMOD1	0--- -x00	0--- -100	0--- -x00	u--- -uuu
SMOD	000- 0011	000- 0011	000- 0011	uuu- uuuu
INTEG	---- 0000	---- 0000	---- 0000	---- uuuu
OPAC0	000- -000	000- -000	000- -000	uuu- -uuu
OPAC1	---0 0000	---0 0000	---0 0000	---u uuuu
ACFC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACFC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFIO0	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFIO1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	-111 1111	-111 1111	-111 1111	-uuu uuuu
PBC	-111 1111	-111 1111	-111 1111	-uuu uuuu
PBPU	-000 0000	-000 0000	-000 0000	-uuu uuuu
PBPD	-000 0000	-000 0000	-000 0000	-uuu uuuu
EEA	---0 0000	---0 0000	---0 0000	---u uuuu
EED	0000 0000	0000 0000	0000 0000	uuuu uuuu
LULV	0000 ----	0000 ----	0000 ----	uuuu ----
HULV	0000 0000	0000 0000	0000 0000	uuuu uuuu
LLL	0000 ----	0000 ----	0000 ----	uuuu ----
HLLV	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEC	---- 0000	---- 0000	---- 0000	---- uuuu
STM1C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1DH	---- --00	---- --00	---- --00	---- --uu
STM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1AH	---- --00	---- --00	---- --00	---- --uu
SIMC0	1110 0000	1110 0000	1110 0000	uuuu uuuu
SIMC1 (UMD=0)	1000 0001	1000 0001	1000 0001	uuuu uuuu
UUCR1* (UMD=1)	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu

Register	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
SIMC2/SIMA/UUCR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMD/UTXR_RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMTOC (UMD=0)	0000 0000	0000 0000	0000 0000	uuuu uuuu
UBRG* (UMD=1)	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
UUSR	0000 1011	0000 1011	0000 1011	uuuu uuuu

Note: “u” stands for unchanged

“x” stands for unknown

“-” stands for unimplemented

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

## Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	—	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	—	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PBPD	—	PBPD6	PBPD5	PBPD4	PBPD3	PBPD2	PBPD1	PBPD0

“—”: Unimplemented, read as “0”

### I/O Logic Function Register List

## Pull-high and Pull-low Resistors

Many product applications require pull-high resistors or pull-low resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor or pull-low resistor. These pull-high resistors are selected using registers PAPU~PBPU and are implemented using weak PMOS transistors. The pull-low resistors in this device are selected using register PBPD and are implemented using weak NMOS transistors.

Note that the pull-high resistor can be controlled by the relevant pull-high control register only when

the pin-shared functional pin is selected as a digital input or NMOS output. Otherwise, the pull-high resistors cannot be enabled. For the pull-low resistors, they can be controlled by the relevant pull-low control register only when the pin-shared functional pin is selected as a digital input, otherwise the pull-low 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 or B, However, the actual available bits for each I/O Port may be different.

• **PBPD Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PBPD6	PBPD5	PBPD4	PBPD3	PBPD2	PBPD1	PBPD0
R/W	—	R/W						
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as “0”

Bit 6~0 **PBPD6~PBPD0:** Port B pin pull-low function control

0: Disable

1: Enable

It should be noted that the pull-high and pull-low resistors cannot be enabled at the same time.

**Port A Wake-up**

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

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

• **PAWU Register**

Bit	7	6	5	4	3	2	1	0
Name	PAWU7	PAWU6	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:** 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~PBC, to control the input/output

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

• **PxC Register**

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

**PxCn:** I/O Port x Pin type selection

- 0: Output
- 1: Input

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

**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 PXS<sub>n</sub>, which can select the desired functions of the multi-function pin-shared pins.

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	—	—	—	—	—	—	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	—	—	—	—	PBS13	PBS12	PBS11	PBS10

**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:** PA6 Pin-shared function selection  
00/01/10: PA6  
11: SDI/SDA/RX
- Bit 5~4     **PAS05~PAS04:** PA5 Pin-shared function selection  
00/01/10: PA5  
11: SDO/TX
- Bit 3~2     **PAS03~PAS02:** PA4 Pin-shared function selection  
00/01/10: PA4/STP0I  
11: A1O
- Bit 1~0     **PAS01~PAS00:** PA3 Pin-shared function selection  
00/01/10: PA3/INT1/STCK0  
11: SCS

• **PAS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PAS11	PAS10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

- Bit 7~2     Unimplemented, read as “0”
- Bit 1~0     **PAS11~PAS10:** PA7 Pin-shared function selection  
00/01/10: PA7  
11: SCK/SCL

• **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  
00/01/10: PB3  
11: AN3
- Bit 5~4     **PBS05~PBS04:** PB2 Pin-shared function selection  
00/01/10: PB2  
11: AN2
- Bit 3~2     **PBS03~PBS02:** PB1 Pin-shared function selection  
00/01/10: PB1

11: AN1  
Bit 1~0 **PBS01~PBS00**: PB0 Pin-shared function selection  
00/01/10: PB0  
11: AN0

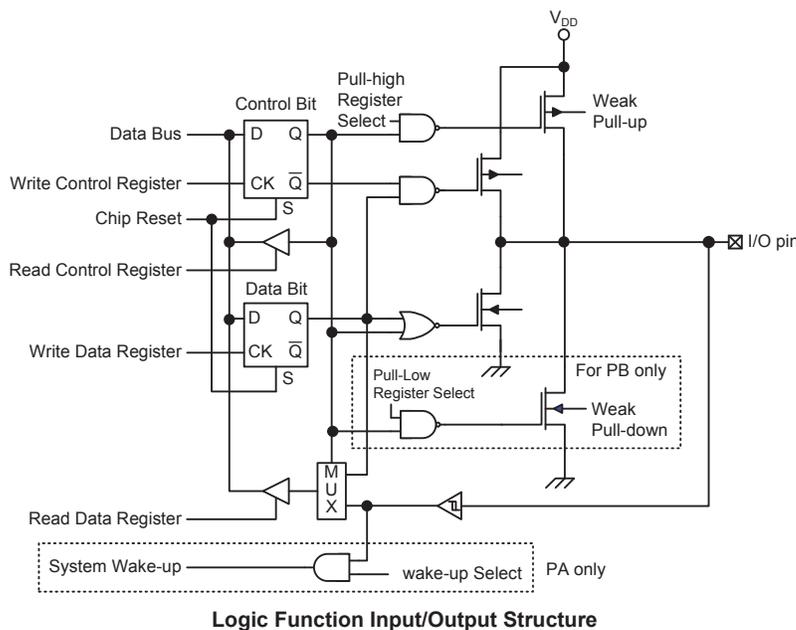
• **PBS1 Register**

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  
00/01/10: PB5  
11: STP1
- Bit 1~0 **PBS11~PBS10:** PB4 Pin-shared function selection  
00/01/10: PB4  
11: STP0

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

place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

Port A has the additional capability of providing wake-up 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, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

The brief features of the Standard type TMs are described here with more detailed information provided in the individual Standard Type TM sections.

### Introduction

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

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

**TM Function Summary**

### TM Operation

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

### TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the STnCK2~STnCK0 bits in the STMn control registers, where “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 STCKn pin. The STCKn pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

### TM Interrupts

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

### TM External Pins

Each of the TMs has a TM input pin, with the label STCKn. The STMn input pin, STCKn, is essentially a clock source for the STMn and is selected using the STnCK2~STnCK0 bits in the STMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The STCKn input pin can be chosen to have either a rising or falling active edge.

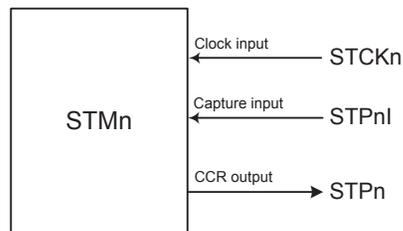
The Standard type TM has another input pin, STPnI, which is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the STnIO1~STnIO0 bits in the STMnC1 register.

The TMs each has one output pins, STPn. The TM output pin can be selected using the corresponding pin-shared function selection bits described in the Pin-shared Function section. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external STPn 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 register. The details of the pin-shared function selection are described in the pin-shared function section.

STM	
Input	Output
STCKn, STPnI	STPn

TM External Pins

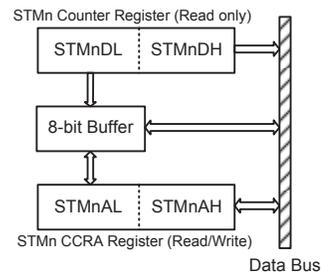


STMn Function Pin Block Diagram (n=0~1)

## Programming Considerations

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

As the CCRA register is 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 low byte register, named STMnAL, using the following access procedures. Accessing the CCRA low byte registers without following these access procedures will result in unpredictable values.

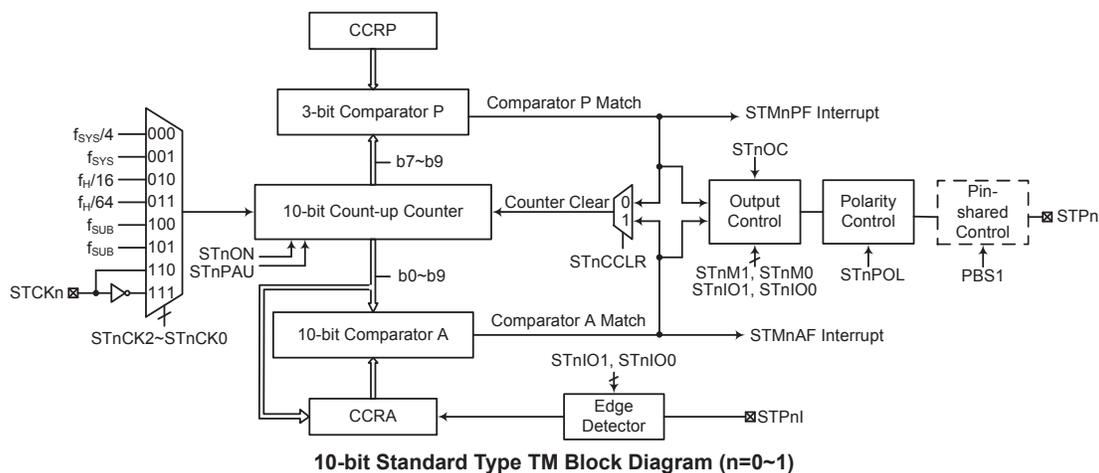


The following steps show the read and write procedures:

- Writing Data to CCRA
  - ♦ Step 1. Write data to Low Byte STMnAL
    - Note that here data is only written to the 8-bit buffer.
  - ♦ Step 2. Write data to High Byte STMnAH
    - Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA
  - ♦ Step 1. Read data from the High Byte STMnDH, STMnAH
    - 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 STMnDL, STMnAL
    - This step reads data from the 8-bit buffer.

## Standard Type TM – STM

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



### Standard Type TM Operation

The size of Standard 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 comparator is 3-bit wide whose value is compared with the highest 3 bits in the counter while the CCRA is the 10 bits and therefore compares 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 STnON 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 STMn interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control an output pin. All operating setup conditions are selected using relevant internal registers.

### Standard Type TM Register Description

Overall operation of the Standard TM is controlled using a series of registers. A read only register pair exists to store the internal counter 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 3-bit CCRP value.

Register Name	Bit							
	7	6	5	4	3	2	1	0
STMnC0	STnPAU	STnCK2	STnCK1	STnCK0	STnON	STnRP2	STnRP1	STnRP0
STMnC1	STnM1	STnM0	STnIO1	STnIO0	STnOC	STnPOL	STnDPX	STnCCLR
STMnDL	D7	D6	D5	D4	D3	D2	D1	D0
STMnDH	—	—	—	—	—	—	D9	D8
STMnAL	D7	D6	D5	D4	D3	D2	D1	D0
STMnAH	—	—	—	—	—	—	D9	D8

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

• **STMnDL 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      STMn Counter Low Byte Register bit 7 ~ bit 0  
 STMn 10-bit Counter bit 7 ~ bit 0

• **STMnDH 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      STMn Counter High Byte Register bit 1 ~ bit 0  
 STMn 10-bit Counter bit 9 ~ bit 8

• **STMnAL 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      STMn CCRA Low Byte Register bit 7 ~ bit 0  
 STMn 10-bit CCRA bit 7 ~ bit 0

• **STMnAH 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      STMn CCRA High Byte Register bit 1 ~ bit 0  
 STMn 10-bit CCRA bit 9 ~ bit 8

• **STMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	STnPAU	STnCK2	STnCK1	STnCK0	STnON	STnRP2	STnRP1	STnRP0
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      **STnPAU**: STMn 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 STMn 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     **STnCK2~STnCK0**: Select STMn Counter clock  
                   000:  $f_{SYS}/4$   
                   001:  $f_{SYS}$   
                   010:  $f_H/16$   
                   011:  $f_H/64$   
                   100:  $f_{SUB}$   
                   101:  $f_{SUB}$   
                   110: STCKn rising edge clock  
                   111: STCKn falling edge clock

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

Bit 3         **STnON**: STMn Counter On/Off control  
                   0: Off  
                   1: On

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

Bit 2~0     **STnRP2~STnRP0**: STMn CCRP 3-bit register, compared with the STMn Counter bit 9 ~ bit 7  
                   Comparator P Match Period  
                   000: 1024 STMn clocks  
                   001: 128 STMn clocks  
                   010: 256 STMn clocks  
                   011: 384 STMn clocks  
                   100: 512 STMn clocks  
                   101: 640 STMn clocks  
                   110: 768 STMn clocks  
                   111: 896 STMn 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 STnCCLR bit is set to zero. Setting the STnCCLR 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.

• **STMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	STnM1	STnM0	STnIO1	STnIO0	STnOC	STnPOL	STnDPX	STnCCLR
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 **STnM1~STnM0**: Select STMn Operating Mode  
 00: Compare Match Output Mode  
 01: Capture Input Mode  
 10: PWM Output Mode or Single Pulse Output Mode  
 11: Timer/Counter Mode

These bits setup the required operating mode for the STMn. To ensure reliable operation the STMn should be switched off before any changes are made to the STnM1 and STnM0 bits. In the Timer/Counter Mode, the STMn output pin state is undefined.

Bit 5~4 **STnIO1~STnIO0**: Select STMn external pin STPn function

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

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

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

Timer/Counter Mode  
 Unused

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

In the Compare Match Output Mode, the STnIO1 and STnIO0 bits determine how the STMn output pin changes state when a compare match occurs from the Comparator A. The TM 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 STMn output pin should be setup using the STnOC bit in the STMnC1 register. Note that the output level requested by the STnIO1 and STnIO0 bits must be different from the initial value setup using the STnOC bit otherwise no change will occur on the STMn output pin when a compare match occurs. After the STMn output pin changes state, it can be reset to its initial level by changing the level of the STnON bit from low to high.

In the PWM Output Mode, the STnIO1 and STnIO0 bits determine how the STMn 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 STnIO1 and STnIO0 bits only after the STMn has been switched off. Unpredictable PWM outputs will occur if the STnIO1 and STnIO0 bits are changed when the STMn is running.

- Bit 3      **STnOC:** STMn STPn Output control  
 Compare Match Output Mode  
     0: Initial low  
     1: Initial high  
 PWM Output Mode/Single Pulse Output Mode  
     0: Active low  
     1: Active high  
 This is the output control bit for the STMn output pin. Its operation depends upon whether STM is being used in the Compare Match Output Mode or in the PWM Output Mode/Single Pulse Output Mode. It has no effect if the STM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the STMn 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 STMn output pin when the STnON bit changes from low to high.
- Bit 2      **STnPOL:** STMn STPn Output polarity control  
     0: Non-invert  
     1: Invert  
 This bit controls the polarity of the STPn output pin. When the bit is set high the STMn output pin will be inverted and not inverted when the bit is zero. It has no effect if the STMn is in the Timer/Counter Mode.
- Bit 1      **STnDPX:** STMn PWM duty/period control  
     0: CCRP – period; CCRA – duty  
     1: CCRP – duty; CCRA – period  
 This bit determines which of the CCRA and CCRP registers are used for period and duty control of the PWM waveform.
- Bit 0      **STnCCLR:** STMn Counter Clear condition selection  
     0: Comparator P match  
     1: Comparator A match  
 This bit is used to select the method which clears the counter. Remember that the Standard Type TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the STnCCLR 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 STnCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

## Standard Type TM Operation Modes

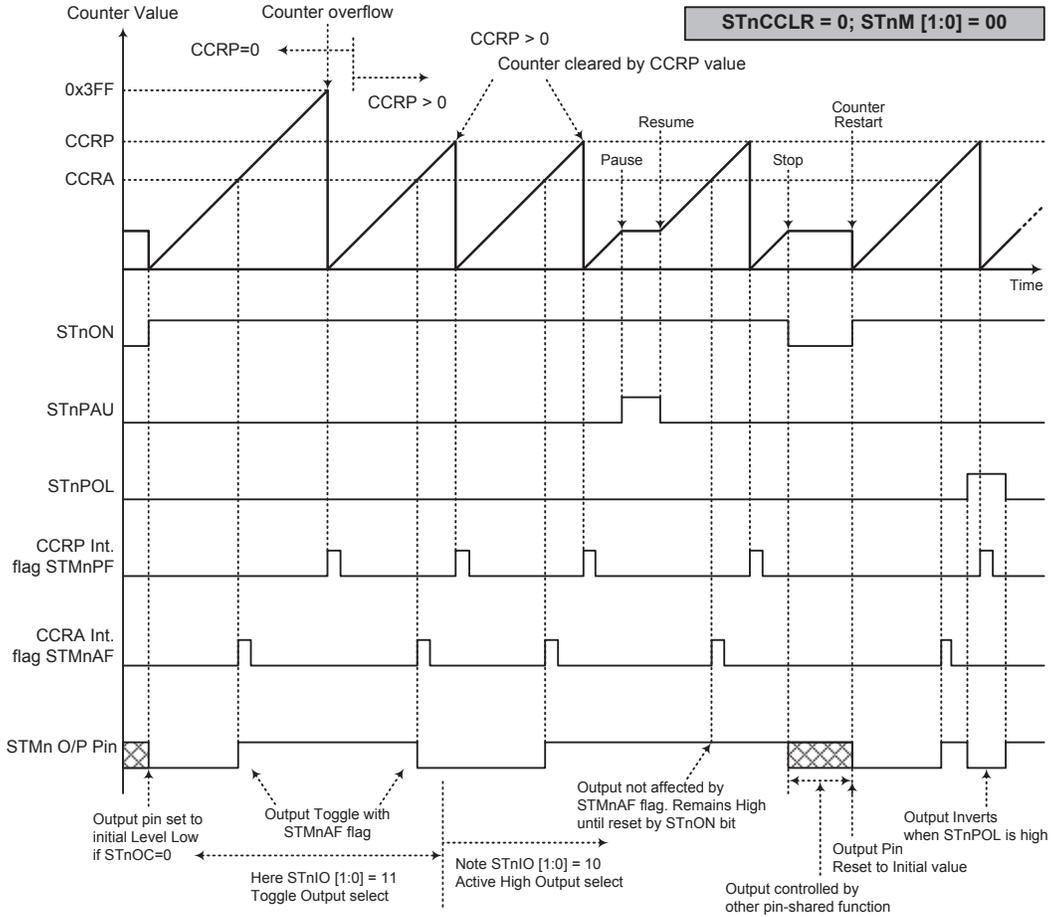
The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STnM1 and STnM0 bits in the STMnC1 register.

### Compare Match Output Mode

To select this mode, bits STnM1 and STnM0 in the STMnC1 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 STnCCLR 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 STMnAF and STMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

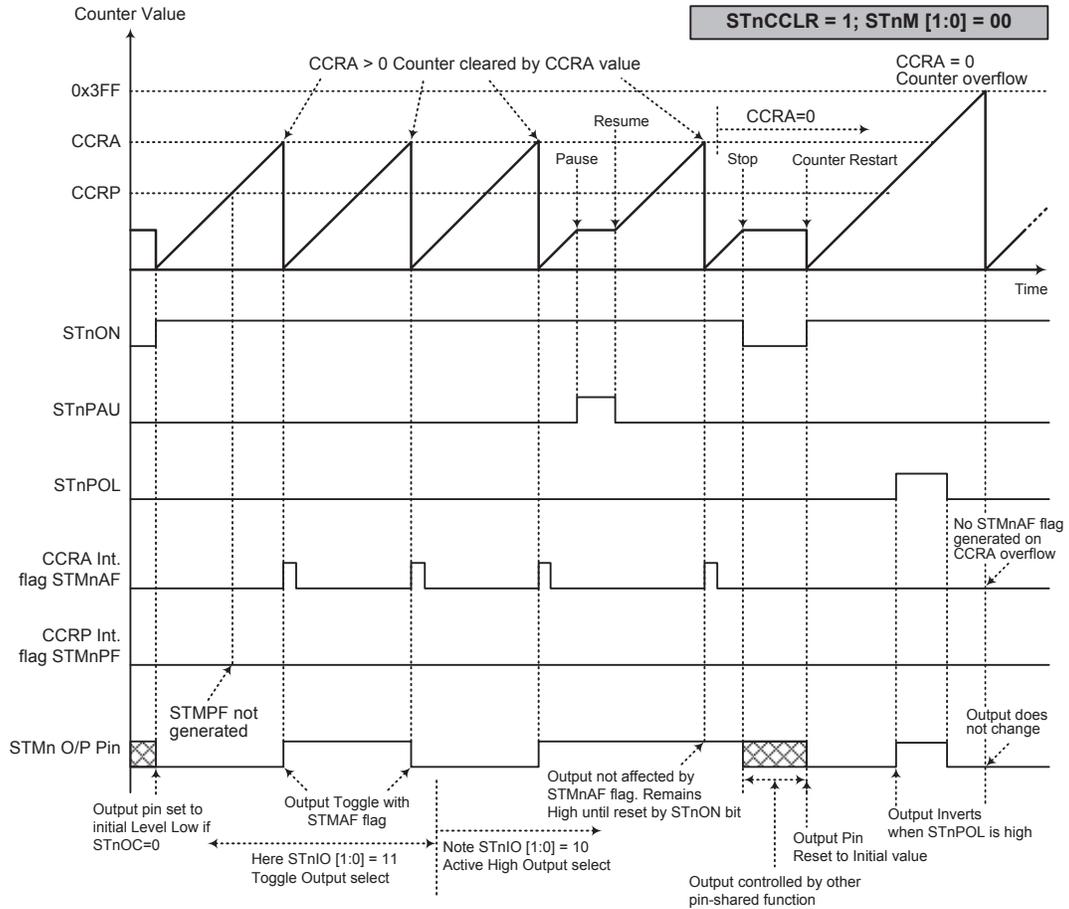
If the STnCCLR bit in the STMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when STnCCLR is high no STMnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to "0". If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the STMnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the STMn output pin, will change state. The STMn output pin condition however only changes state when a STMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The STMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the STMn output pin. The way in which the STMn output pin changes state are determined by the condition of the STnIO1 and STnIO0 bits in the STMnC1 register. The STMn output pin can be selected using the STnIO1 and STnIO0 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 STMn output pin, which is setup after the STnON bit changes from low to high, is setup using the STnOC bit. Note that if the STnIO1 and STnIO0 bits are zero then no pin change will take place.



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

- Note: 1. With  $STnCCR=0$  a Comparator P match will clear the counter  
 2. The STMn output pin is controlled only by the STMnAF flag  
 3. The output pin is reset to its initial state by an STnON bit rising edge



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

- Note: 1. With STnCCLR=1 a Comparator A match will clear the counter
2. The STMn output pin is controlled only by the STMnAF flag
3. The output pin is reset to its initial state by an STnON bit rising edge
4. An STMnPF flag is not generated when STnCCLR=1

### Timer/Counter Mode

To select this mode, bits STnM1 and STnM0 in the STMnC1 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 STMn 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 STMn 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 STnM1 and STnM0 in the STMnC1 register should be set to 10 respectively and also the STnIO1 and STnIO0 bits should be set to 10 respectively. The PWM function within the STMn 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 STMn 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 STnCCLR bit has no effect as the PWM period. 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 STnDPX bit in the STMnC1 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 STnOC bit in the STMnC1 register is used to select the required polarity of the PWM waveform while the two STnIO1 and STnIO0 bits are used to enable the PWM output or to force the STMn output pin to a fixed high or low level. The STnPOL bit is used to reverse the polarity of the PWM output waveform.

• **10-bit STM, PWM Output Mode, Edge-aligned Mode, STnDPX=0**

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

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

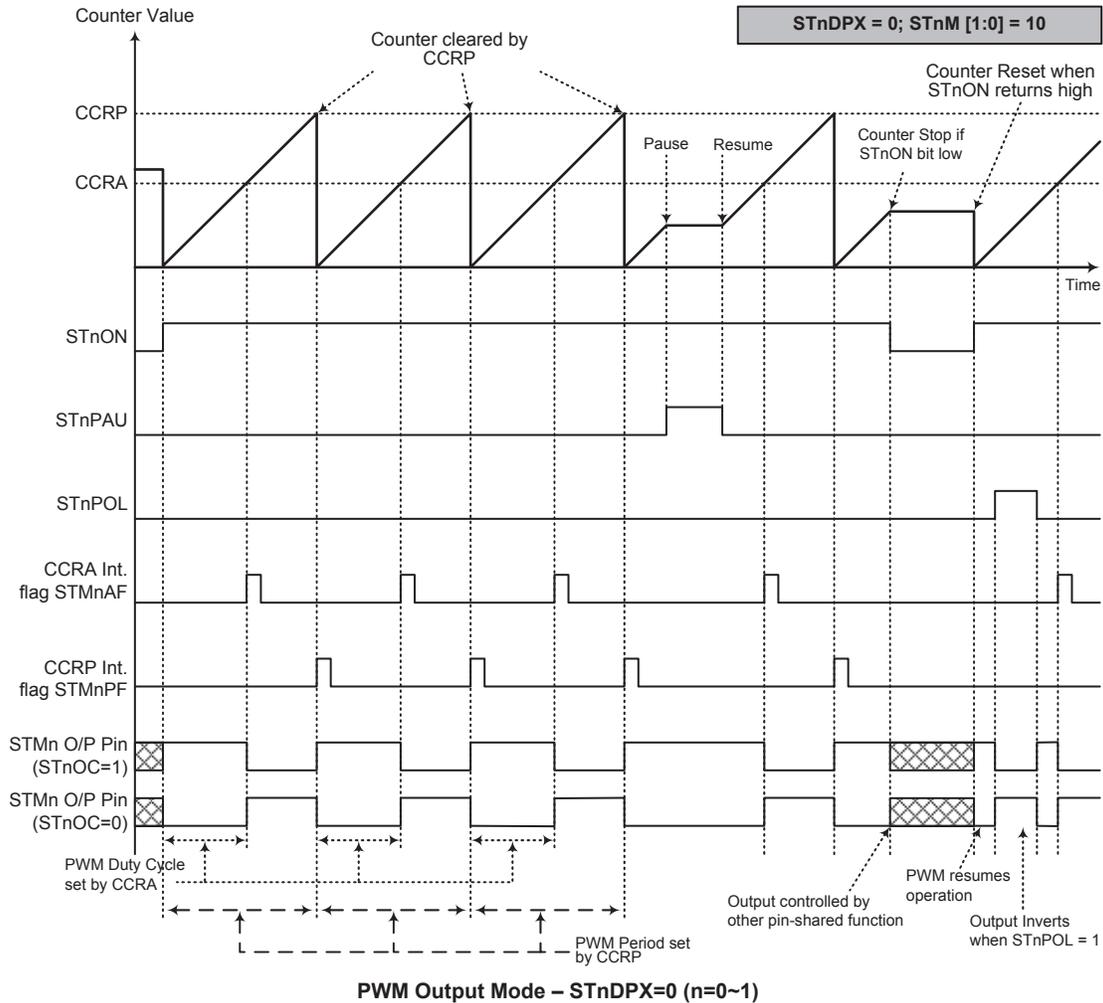
The STM PWM output frequency= $(f_{SYS}/4)/(2 \times 128)=f_{SYS}/1024=4\text{kHz}$ , duty= $128/(2 \times 128)=50\%$ .

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

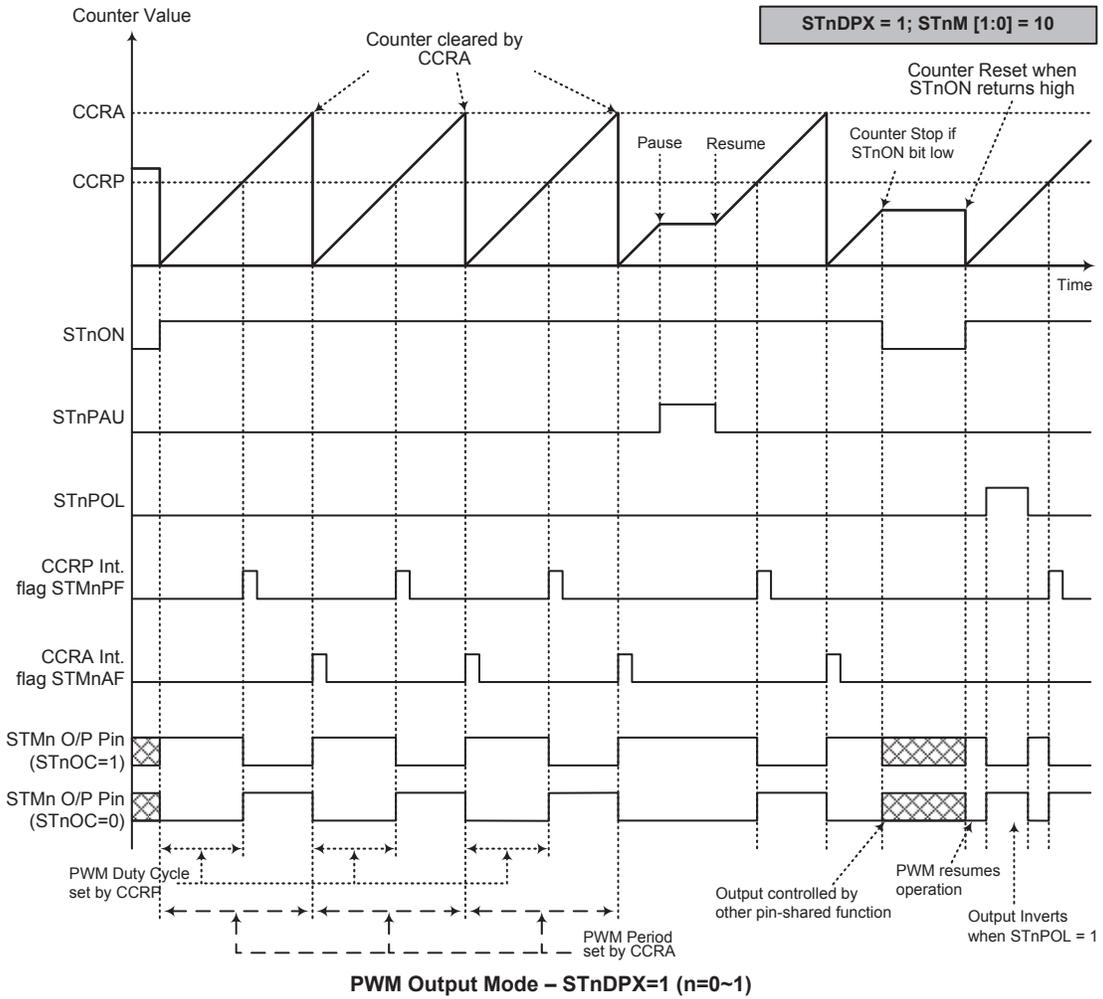
• **10-bit STM, PWM Output Mode, Edge-aligned Mode, STnDPX=1**

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

The PWM output period is determined by the CCRA register value together with the STMn clock while the PWM duty cycle is defined by the CCRP register value except when the CCRP value is equal to 0.



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



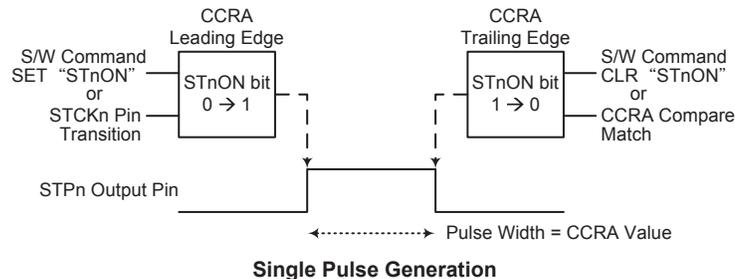
- Note: 1. Here  $STnDPX=1$  – Counter cleared by CCRA  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues even when  $STnIO [1:0]=00$  or  $01$   
 4. The  $STnCCLR$  bit has no influence on PWM operation

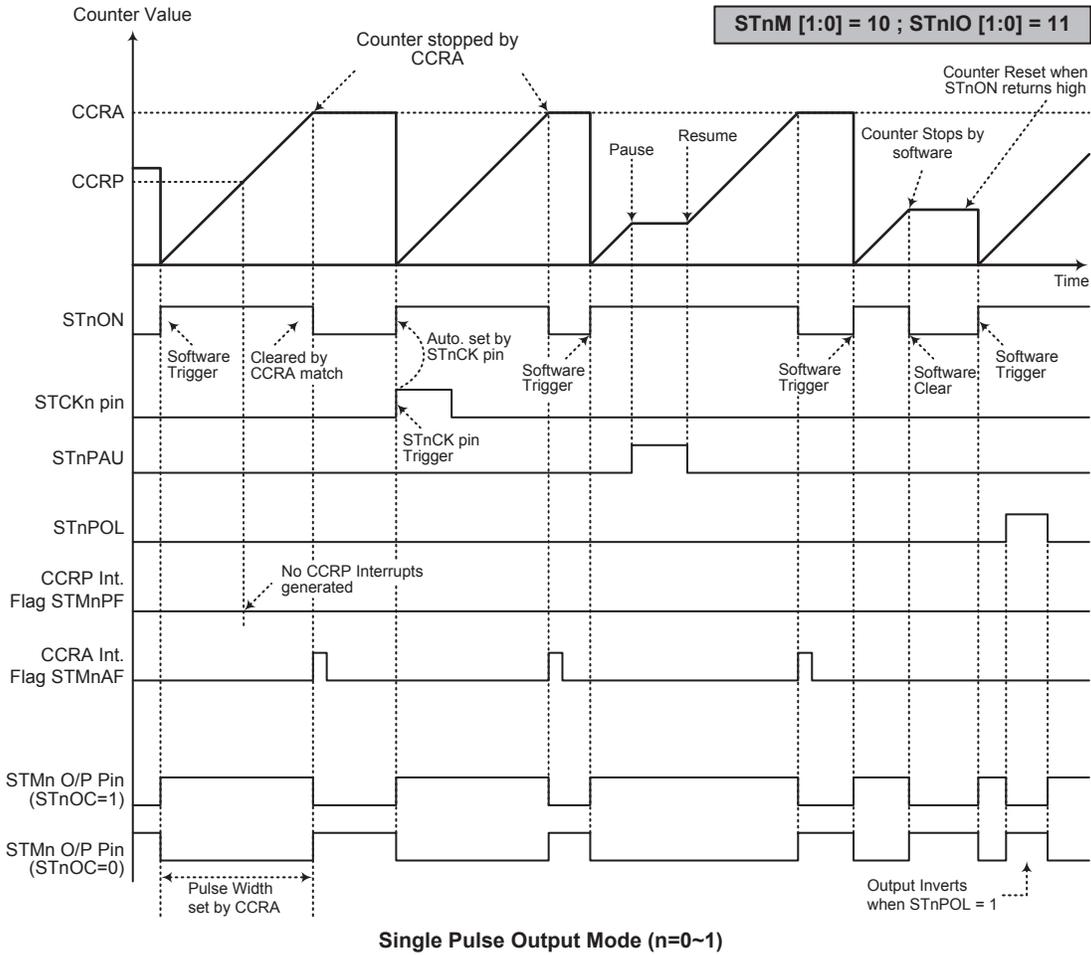
**Single Pulse Output Mode**

To select this mode, bits STnM1 and STnM0 in the STMnC1 register should be set to 10 respectively and also the STnIO1 and STnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STMn output pin.

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



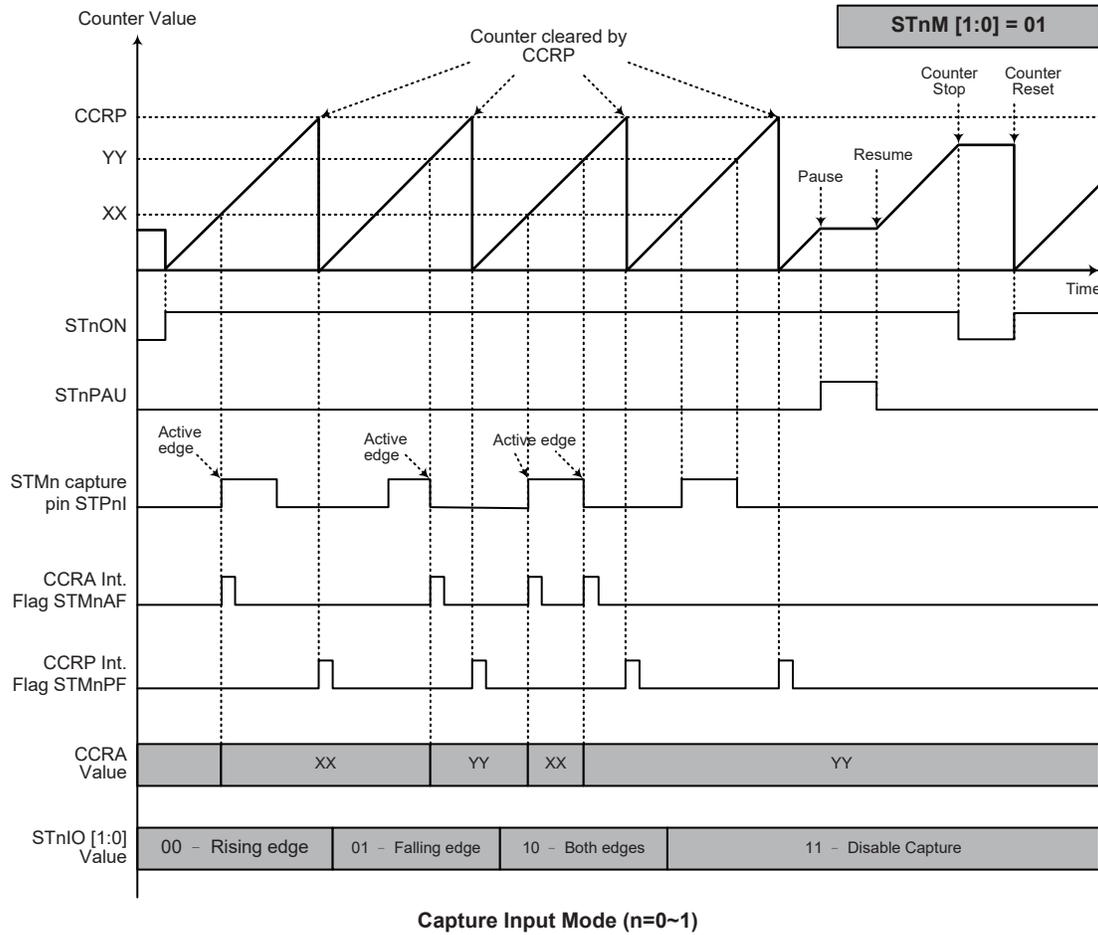


- Note:
1. Counter stopped by CCRA
  2. CCRP is not used
  3. The pulse triggered by the STnCK pin or by setting the STnON bit high
  4. An STCKn pin active edge will automatically set the STnON bit high
  5. In the Single Pulse Output Mode, STnIO [1:0] must be set to "11" and can not be changed

### **Capture Input Mode**

To select this mode bits STnM1 and STnM0 in the STMnC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the STPnI pin, whose active edge can be a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the STnIO1 and STnIO0 bits in the STMnC1 register. The counter is started when the STnON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the STPnI pin the present value in the counter will be latched into the CCRA registers and an STMn interrupt generated. Irrespective of what events occur on the STPnI pin the counter will continue to free run until the STnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, an STMn interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The STnIO1 and STnIO0 bits can select the active trigger edge on the STPnI pin to be a rising edge, falling edge or both edge types. If the STnIO1 and STnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the STPnI pin, however it must be noted that the counter will continue to run. The STnCCLR and STnDPX bits are not used in this Mode.



- Note: 1. STnM [1:0]=01 and active edge set by the STnIO [1:0] bits
2. An STMn Capture input pin active edge transfers the counter value to CCRA
3. STnCCLR bit not used
4. No output function – STnOC and STnPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero

## Analog to Digital Converter – ADC

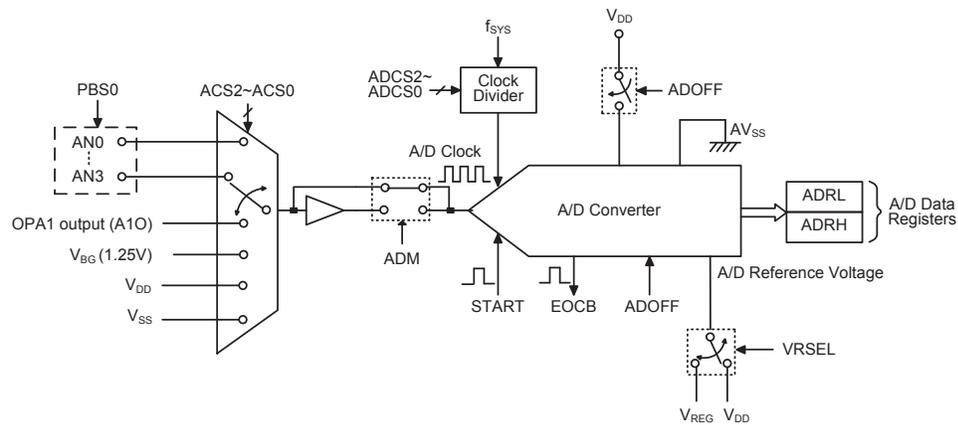
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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 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.

Input Channels	A/D Channel Select Bits	External Input Pins	Internal Input Signals
4+4	ACS2~ACS0	AN0~AN3	A1O, V <sub>BG</sub> , V <sub>DD</sub> , V <sub>SS</sub>

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



**A/D Converter Structure**

### A/D Converter Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
ADCR0	START	EOCB	ADOFF	ADM	D3	ACS2	ACS1	ACS0
ADCR1	TEST	—	—	VRSEL	OPA1V	ADCS2	ADCS1	ADCS0
ADRL	D3	D2	D1	D0	—	—	—	—
ADRH	D11	D10	D9	D8	D7	D6	D5	D4

**A/D Converter Register List**

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

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 ADRH, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero. Note that the A/D converter data register contents will be cleared to zero if the A/D converter is disabled.

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

**A/D Data Registers**

#### • ADRL Register

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

Bit 7~4      Lower byte of ADC conversion data

Bit 3~0      Unimplemented, read as “0”

#### • ADRH Register

Bit	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	D7	D6	D5	D4
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0      Higher byte of ADC conversion data

### A/D Converter Control Registers – ADCR0, ADCR1

To control the function and operation of the A/D converter, two control registers, known as ADCR0 and ADCR1, are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The ACS2~ACS0 bits in the ADCR0 register are used to determine which channel input is selected to be converted.

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.

• **ADCR0 Register**

Bit	7	6	5	4	3	2	1	0
Name	START	EOCB	ADOFF	ADM	D3	ACS2	ACS1	ACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	1	0	0	0	0	0

- Bit 7     **START**: Start the A/D conversion  
0 → 1 → 0: Start A/D conversion  
0 → 1: Reset the A/D converter and set EOCB to 1  
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. When the bit is set high the A/D converter will be reset.
- Bit 6     **EOCB**: End of A/D conversion flag  
0: A/D conversion ended  
1: A/D conversion is in progress  
This read only flag is used to indicate when an A/D conversion process has completed. When the conversion process is running, the bit will be high.
- Bit 5     **ADOFF**: ADC module on/off control bit  
0: ADC module is enabled  
1: ADC module is disabled  
This bit controls the A/D internal function. This bit should be cleared to zero to enable the A/D converter. If the bit is set high, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair, ADRH and ADRL, will be cleared to zero.
- Bit 4     **ADM**: A/D Converter mode selection  
0: Normal mode (analog input bypass pre-buffer, direct to ADC)  
1: High drive mode (analog input through pre-buffer to ADC)  
When ADM=1 and ADOFF=0, the analog signal will be through buffer to A/D converter.
- Bit 3     **D3**: Data bit, can be read or written.
- Bit 2~0   **ACS2~ACS0**: A/D converter input channel selection  
000: External channel – AN0  
001: External channel – AN1  
010: External channel – AN2  
011: External channel – AN3  
100: Internal source – OPA1 output, A10  
101: Internal source – V<sub>BG</sub>  
110: Internal source – V<sub>DD</sub>  
111: Internal source – V<sub>SS</sub>

• **ADCR1 Register**

Bit	7	6	5	4	3	2	1	0
Name	TEST	—	—	VRSEL	OPA1V	ADCS2	ADCS1	ADCS0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	1	—	—	0	0	0	0	0

- Bit 7     **TEST**: For test only, read always as “1”
- Bit 6~5   Unimplemented, read as “0”
- Bit 4     **VRSEL**: A/D converter reference voltage selection  
0: V<sub>DD</sub>  
1: V<sub>REG</sub>

- Bit 3      **OPA1V**: OPA1 power voltage selection  
             0: V<sub>DD</sub>  
             1: LDO
- Bit 2~0    **ADCS2~ADCS0**: A/D converter clock source selection  
             000: f<sub>sys</sub>/2  
             001: f<sub>sys</sub>/8  
             010: f<sub>sys</sub>/32  
             011: Undefined  
             100: f<sub>sys</sub>  
             101: f<sub>sys</sub>/4  
             110: f<sub>sys</sub>/16  
             111: Undefined

### A/D Operation

The START bit is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the EOCB bit in the ADCR0 register will be cleared to zero and the analog to digital converter will be reset. It is the START bit that is used to control the overall start operation of the internal analog to digital converter.

The EOCB bit in the ADCR0 register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to “0” by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the EOCB bit in the ADCR0 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<sub>sys</sub>, can be chosen to be either f<sub>sys</sub> or a subdivided version of f<sub>sys</sub>. The division ratio value is determined by the ADCS2~ADCS0 bits in the ADCR1 register. Although the A/D clock source is determined by the system clock f<sub>sys</sub>, and by bits ADCS2~ADCS0, there are some limitations on the A/D clock source speed that can be selected. As the recommended value of permissible A/D clock period, t<sub>ADCK</sub>, is from 0.5μs to 10μs, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the ADCS2~ADCS0 bits should not be set to “100”. 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, depending upon the device, special care must be taken, as the values may be less than or greater than the specified A/D Clock Period.

f <sub>sys</sub>	A/D Clock Period (t <sub>ADCK</sub> )						
	ADCS2, ADCS1, ADCS0 =100 (f <sub>sys</sub> )	ADCS2, ADCS1, ADCS0 =000 (f <sub>sys</sub> /2)	ADCS2, ADCS1, ADCS0 =101 (f <sub>sys</sub> /4)	ADCS2, ADCS1, ADCS0 =001 (f <sub>sys</sub> /8)	ADCS2, ADCS1, ADCS0 =110 (f <sub>sys</sub> /16)	ADCS2, ADCS1, ADCS0 =010 (f <sub>sys</sub> /32)	ADCS2, ADCS1, ADCS0 =011, 111
1MHz	1μs	2μs	4μs	8μs	16μs*	32μs*	Undefined
2MHz	500ns	1μs	2μs	4μs	8μs	16μs*	Undefined
4MHz	250ns*	500ns	1μs	2μs	4μs	8μs	Undefined
8MHz	125ns*	250ns*	500ns	1μs	2μs	4μs	Undefined

**A/D Clock Period Examples**

Controlling the power on/off function of the A/D converter circuitry is implemented using the ADOFF bit in the ADCR0 register. This bit must be cleared to zero to power on the A/D converter. When the ADOFF bit is cleared to zero 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 by configuring the corresponding pin control bits, if the ADOFF bit is low then some power will still be consumed. In power conscious applications it is therefore recommended that the ADOFF is set high to reduce power consumption when the A/D converter function is not being used.

### A/D Reference Voltage

The reference voltage supply to the A/D Converter can be supplied from the positive power supply,  $V_{DD}$  or an external pin VREG. The desired selection is made using the VRSEL bit in the ADCR1 register

### A/D Converter Input Signal

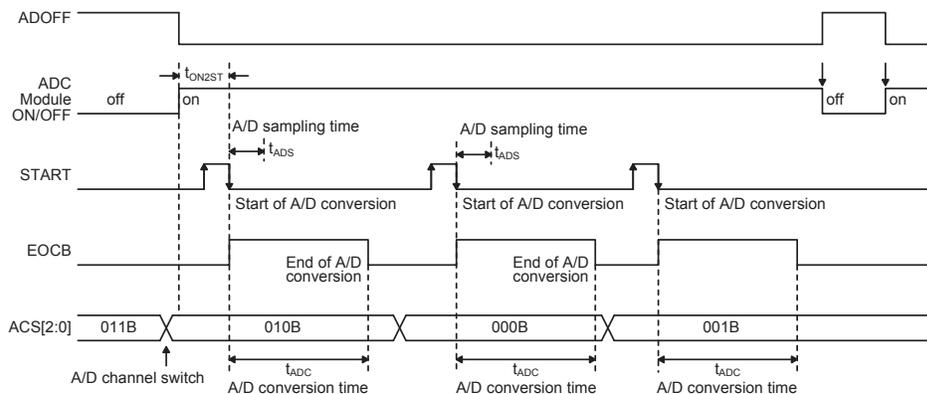
All of the A/D analog input pins are pin-shared with the I/O pins on Port B. The corresponding pin-shared function selection bit in the PBS0 register, determines whether the input pin is setup as A/D converter analog input or whether it has other functions. If the corresponding pin is setup to be an 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 control bits enable an A/D input, the status of the port control register will be overridden.

### Conversion Rate and Timing Diagram

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

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

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



**A/D Conversion Timing**

## Summary of A/D Conversion Steps

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

- Step 1  
Select the required A/D conversion clock by correctly programming bits ADCS2~ADCS0 in the ADCR1 register.
- Step 2  
Enable the A/D converter by clearing the ADOFF bit in the ADCR0 register to zero.
- Step 3  
Select which channel is to be connected to the internal A/D converter by correctly programming the ACS2~ACS0 bits which are also contained in the ADCR0 register
- Step 4  
Select which pins are to be used as A/D inputs and configure them by correctly programming the PBS0 register.
- Step 5  
If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, Mult-function interrupt bit, MFIE, and the A/D converter interrupt bit, ADE, must all be set high to do this.
- Step 6  
The analog to digital conversion process can now be initialised by setting the START bit in the ADCR0 register from low to high and then low again. Note that this bit should have been originally cleared to zero.
- Step 7  
To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR0 register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

Note: When checking for the end of the conversion process, if the method of polling the EOCB bit in the ADCR0 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 ADOFF high in the ADCR0 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  $V_{REF}$  divided by 4096.

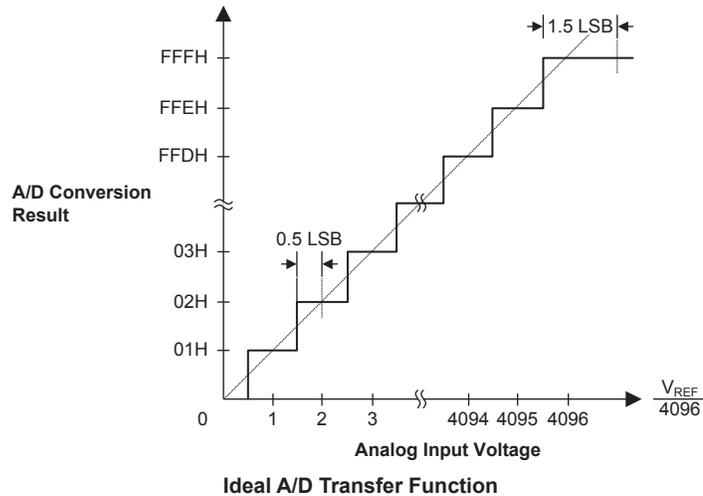
$$1 \text{ LSB} = V_{REF} / 4096$$

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 VRSEL bit.



### 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 EOCB bit in the ADCR0 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 EOCB Polling Method to Detect the end of Conversion

```

clr ADE           ; disable ADC interrupt
mov a,01H
mov ADCR1,a       ; select fsys/8 as A/D clock and select VDD as ADC reference voltage
mov a,03h         ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,00h
mov ADCR0,a       ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr START         ; high pulse on start bit to initiate conversion
set START        ; reset A/D
clr START        ; start A/D
polling_EOC:
sz EOCB          ; poll the ADCR0 register EOCB bit to detect end of A/D conversion
jmp polling_EOC  ; continue polling
mov a,ADRL       ; read low byte conversion result value
mov ADRL_buffer,a ; save result to user defined register
mov a,ADRH       ; read high byte conversion result value
mov ADRH_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,01H
mov ADCR1,a      ; select fsys/8 as A/D clock and select VDD as ADC reference voltage
mov a,03h        ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,00h
mov ADCR0,a      ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr START        ; high pulse on START bit to initiate conversion
set START        ; reset A/D
clr START        ; start A/D
clr ADF          ; clear ADC interrupt request flag
set ADE          ; enable ADC interrupt
set MFIE         ; enable Multi-function interrupt 1
set EMI          ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov acc_stack,a  ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a ; save STATUS to user defined memory
:
:
mov a,ADRL       ; read low byte conversion result value
mov ADRL_buffer,a ; save result to user defined register
mov a,ADRH       ; read high byte conversion result value
mov ADRH_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
```

## Auto Conversion Function

The device contains an auto conversion circuit function which is used to enable automatic A/D conversions. This function can be implemented using the WDT counter.

### Auto Conversion Operation

The Auto conversion circuit allows the WDT counter to enable the ADC and to compare the ADC conversion value with pre-programmed upper and lower limit values while in the power down state.

The auto A/D conversion time is selected by the user controlled bits, ACFT2~ACFT0, in the ACFC0 register. When the WDT counts up to this period time, it will send a signal to the ACC circuit, to enable the ADC and automatically start a conversion. If the ADC data does not lie between the preset lower and upper limit values, the event counter value ECNT2~ECNT0 will be increased by one. When the ECNT2~ECNT0 value is equal to the number of events as set by bits NOE1~NOE0, the ACC will send an interrupt signal to MCU and the hardware will clear the ECNT2~ECNT0 bits. The CPU will then be woken up and execute the ACC interrupt subroutine.

### Auto Conversion Registers

To control the operation of the Auto Conversion function, several control registers known as ACFC0, ACFC1, LULV, HULV, LLLV and HLLV are provided. The ACFC0 register is used to enable or disable the auto conversion function and set the auto conversion time. When the WDT counts up to a preset time, the hardware circuit will automatically enable the ADC function. The ACFC1 register is the event counter value register. The remaining four data registers LULV, HULV, LLLV and HLLV are used to store the lower and upper limit values.

#### • ACFC0 Register

Bit	7	6	5	4	3	2	1	0
Name	ACFEN	FH2AD	OP2AD	—	—	ACFT2	ACFT1	ACFT0
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7 **ACFEN**: Auto conversion function control bit

0: Disable  
 1: Enable

Bit 6 **FH2AD**: FH and AD turn-on timing control

0: Enable  
 1: Disable

Bit 5 **OP2AD**: OP and AD turn-on timing control

0: Enable  
 1: Disable

When the FH2AD and OP2AD bits are enabled, the performance of the Auto Conversion Function will be improved.

Bit 4~3 Unimplemented, read as “0”

Bit 2~0 **ACFT2~ACFT0**: Auto A/D conversion time selection

000: 4ms  
 001: 8ms  
 010: 16ms  
 011: 32ms  
 100: 64ms  
 101: 128ms  
 110: 256ms  
 111: 512ms

If the WDT counter value is equal to this period, the hardware circuit will enable the ADC function automatically.

• **ACFC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	ECNT2	ECNT1	ECNT0	—	—	NOE1	NOE0
R/W	—	R/W	R/W	R/W	—	—	R/W	R/W
POR	—	0	0	0	—	—	0	0

Bit 7 Unimplemented, read as “0”

Bit 6~4 **ECNT2~ECNT0**: Event counter values

If the ADC data does not lie between the lower limitation value and upper limitation value, the event counter values ECNT2~ECNT0 will be increased by one. When ECNT2~ECNT0 is equal to the number of events values as set by bits NOE1~NOE0, the ACC circuit will send an interrupt signal to the CPU and the hardware will clear the ECNT2~ECNT0 bits.

Bit 3~2 Unimplemented, read as “0”

Bit 1~0 **NOE1~NOE0**: The number of events

00: 1 time  
01: 2 times  
10: 4 times  
11: 7 times

• **LULV Register**

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

Bit 7~4 Lower byte of upper limitation value

Bit 3~0 Unimplemented, read as “0”

• **HULV Register**

Bit	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	D7	D6	D5	D4
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 Higher byte of upper limitation value

• **LLLV Register**

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

Bit 7~4 Lower byte of Lower limitation value

Bit 3~0 Unimplemented, read as “0”

• **HLLV Register**

Bit	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	D7	D6	D5	D4
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 Higher byte of lower limitation value

## Universal Serial Interface Module – USIM

The device contains a Universal Serial Interface Module, which includes the four-line SPI interface, the two-line I<sup>2</sup>C interface and the two-line UART interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI, I<sup>2</sup>C or UART based hardware such as sensors, Flash or EEPROM memory, etc. The USIM interface pins are pin-shared with other I/O pins therefore the USIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As all the interface types share the same pins and registers, the choice of whether the UART, SPI or I<sup>2</sup>C type is used is made using the UART mode selection bit, named UMD, and the SPI/I<sup>2</sup>C operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the USIM pin-shared I/O are selected using pull-high control registers when the USIM function is enabled and the corresponding pins are used as USIM input pins.

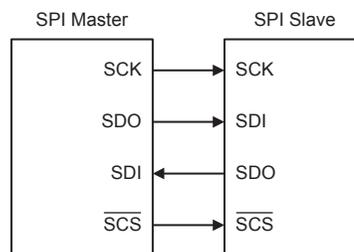
### SPI Interface

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

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

### SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and  $\overline{\text{SCS}}$ . Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and  $\overline{\text{SCS}}$  is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I<sup>2</sup>C/UART function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single  $\overline{\text{SCS}}$  pin only one slave device can be utilized. The  $\overline{\text{SCS}}$  pin is controlled by software, set CSEN bit to 1 to enable  $\overline{\text{SCS}}$  pin function, set CSEN bit to 0 the  $\overline{\text{SCS}}$  pin will be floating state.

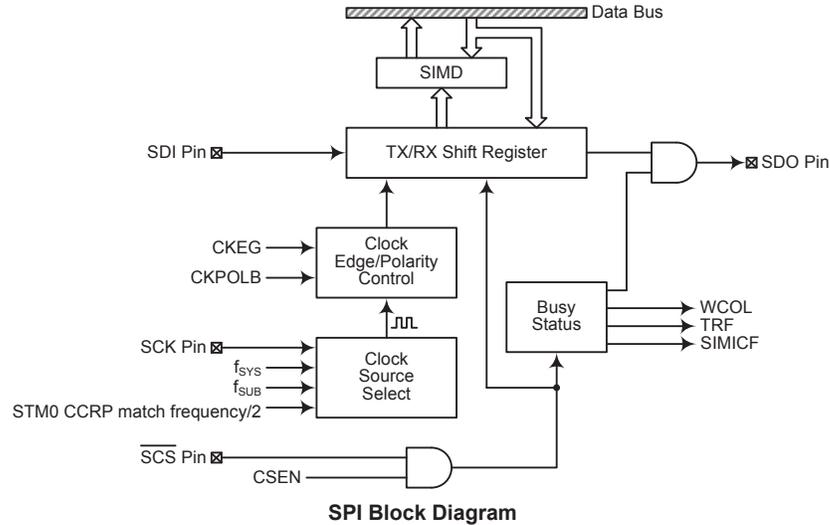


SPI Master/Slave Connection

The SPI function in the device offers the following features:

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

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



### SPI Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
SIMD	D7	D6	D5	D4	D3	D2	D1	D0

**SPI Register List**

**SPI Data Register**

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I<sup>2</sup>C functions. Before the device writes data to the SPI bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the SPI bus, the device can read it from the SIMD register. Any transmission or reception of data from the SPI bus must be made via the SIMD register.

• **SIMD Register**

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

"x": Unknown

Bit 7~0      **D7~D0:** USIM SPI/I<sup>2</sup>C data register bit 7~bit 0

**SPI Control Registers**

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

• **SIMC0 Register**

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

Bit 7~5      **SIM2~SIM0:** USIM SPI/I<sup>2</sup>C Operating Mode Control

- 000: SPI master mode; SPI clock is  $f_{SYS}/4$
- 001: SPI master mode; SPI clock is  $f_{SYS}/16$
- 010: SPI master mode; SPI clock is  $f_{SYS}/64$
- 011: SPI master mode; SPI clock is  $f_{SUB}$
- 100: SPI master mode; SPI clock is STM0 CCRP match frequency/2
- 101: SPI slave mode
- 110: I<sup>2</sup>C slave mode
- 111: Unused mode

When the UMD bit is cleared to zero, these bits setup the SPI or I<sup>2</sup>C operating mode of the USIM function. As well as selecting if the I<sup>2</sup>C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from STM0 and  $f_{SUB}$ . If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

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

- 0: SPI or I<sup>2</sup>C mode
- 1: UART mode

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

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

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

- Bit 1**      **SIMEN:** USIM SPI/I<sup>2</sup>C Enable Control  
                  0: Disable  
                  1: Enable
- The bit is the overall on/off control for the USIM SPI/I<sup>2</sup>C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I<sup>2</sup>C interface, the SDI, SDO, SCK and  $\overline{SCS}$ , or SDA and SCL lines will lose their SPI or I<sup>2</sup>C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I<sup>2</sup>C interface is enabled. If the USIM is configured to operate as an SPI interface via the UMD and SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the USIM is configured to operate as an I<sup>2</sup>C interface via the UMD and SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I<sup>2</sup>C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I<sup>2</sup>C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.
- Bit 0**      **SIMICF:** USIM SPI Incomplete Flag  
                  0: USIM SPI incomplete condition is not occurred  
                  1: USIM SPI incomplete condition is occurred
- This bit is only available when the USIM is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SIMEN and CSEN bits both being set to 1 but the  $\overline{SCS}$  line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set to 1 together with the TRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the TRF bit will not be set to 1 if the SIMICF bit is set to 1 by software application program.

• **SIMC2 Register**

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

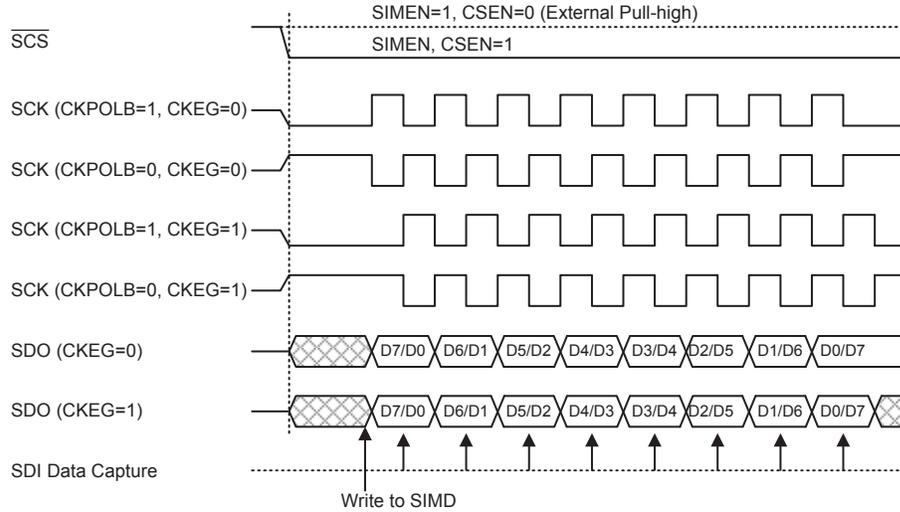
- Bit 7~6**      **D7~D6:** Undefined bits  
 These bits can be read or written by the application program.
- Bit 5**      **CKPOLB:** SPI clock line base condition selection  
                  0: The SCK line will be high when the clock is inactive  
                  1: The SCK line will be low when the clock is inactive
- The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.
- Bit 4**      **CKEG:** SPI SCK clock active edge type selection  
                  CKPOLB=0  
                  0: SCK is high base level and data capture at SCK rising edge  
                  1: SCK is high base level and data capture at SCK falling edge  
                  CKPOLB=1  
                  0: SCK is low base level and data capture at SCK falling edge  
                  1: SCK is low base level and data capture at SCK rising edge
- The CKEG and CKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOLB bit.

- Bit 3      **MLS**: SPI data shift order  
            0: LSB first  
            1: MSB first  
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.
- Bit 2      **CSEN**: SPI  $\overline{SCS}$  pin control  
            0: Disable  
            1: Enable  
The CSEN bit is used as an enable/disable for the  $\overline{SCS}$  pin. If this bit is low, then the  $\overline{SCS}$  pin will be disabled and placed into a floating condition. If the bit is high the  $\overline{SCS}$  pin will be enabled and used as a select pin.
- Bit 1      **WCOL**: SPI write collision flag  
            0: No collision  
            1: Collision  
The WCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program.
- Bit 0      **TRF**: SPI Transmit/Receive complete flag  
            0: SPI data is being transferred  
            1: SPI data transmission is completed  
The TRF bit is the Transmit/Receive Complete flag and is set “1” automatically when an SPI data transmission is completed, but must set to “0” by the application program. It can be used to generate an interrupt.

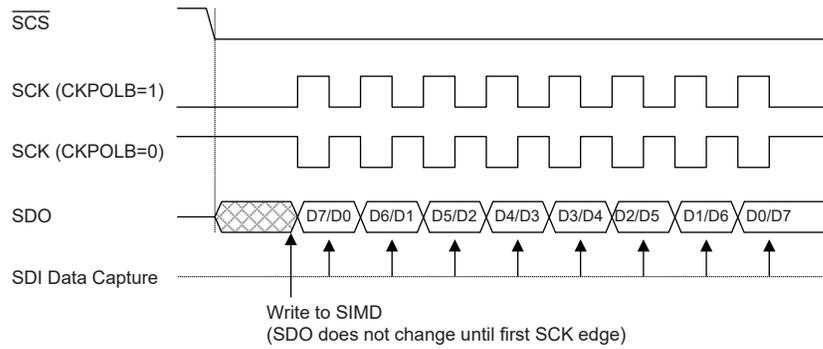
### **SPI Communication**

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

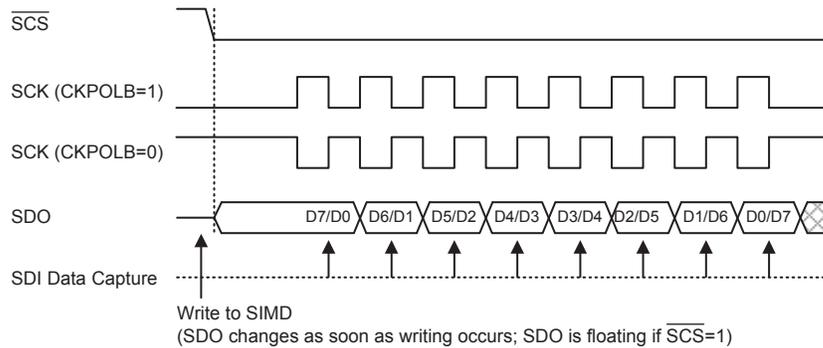
The SPI will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.



**SPI Master Mode Timing**

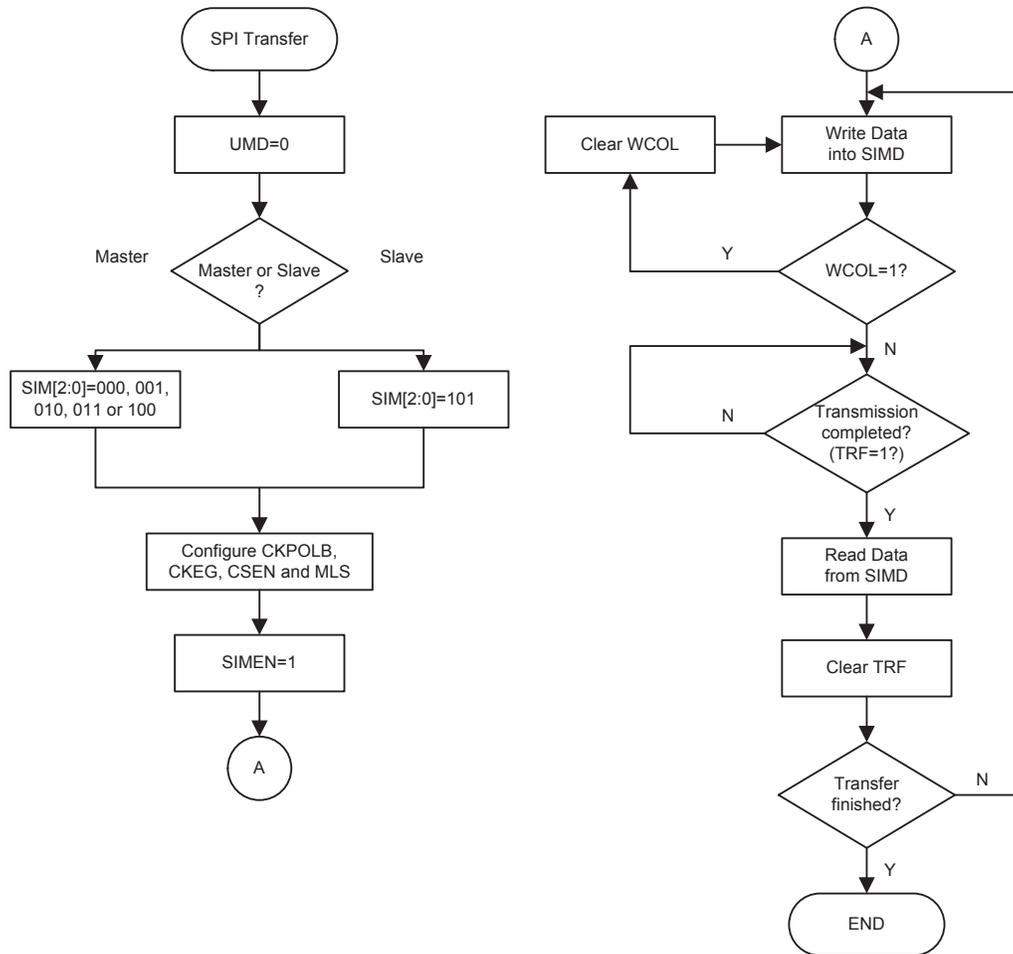


**SPI Slave Mode Timing – CKEG=0**



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

**SPI Slave Mode Timing – CKEG=1**



**SPI Transfer Control Flowchart**

**SPI Bus Enable/Disable**

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

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

**SPI Operation Steps**

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

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

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

### Master Mode

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

### Slave Mode

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

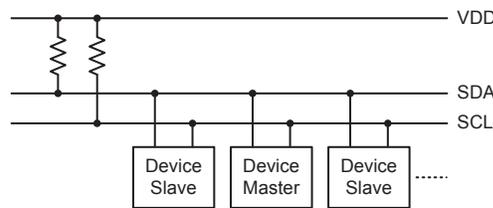
- Step 5  
Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6  
Check the TRF bit or wait for a USIM SPI serial bus interrupt.
- Step 7  
Read data from the SIMD register.
- Step 8  
Clear TRF.
- Step 9  
Go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

### **Error Detection**

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

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

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

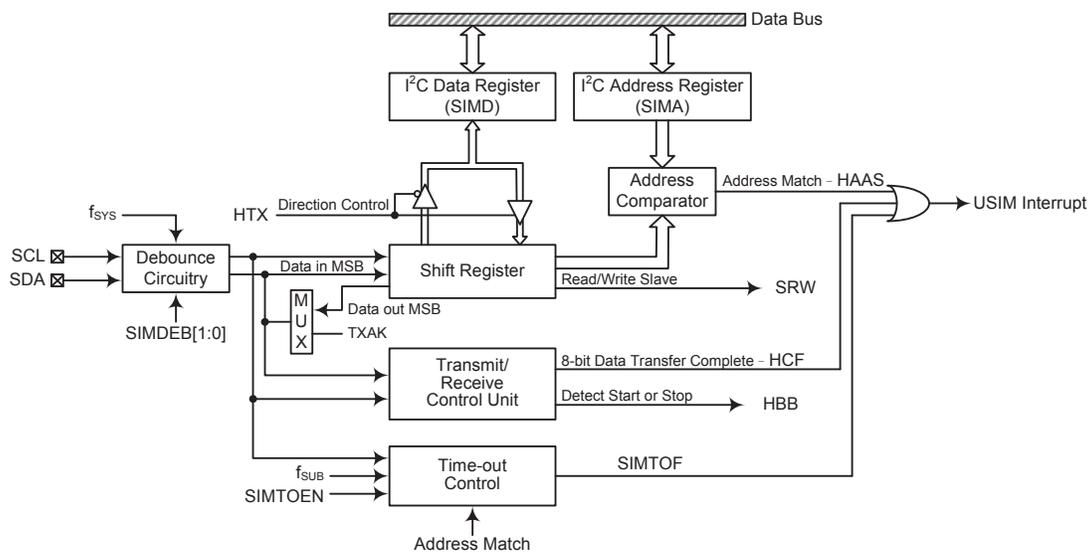


**I<sup>2</sup>C Master Slave Bus Connection**

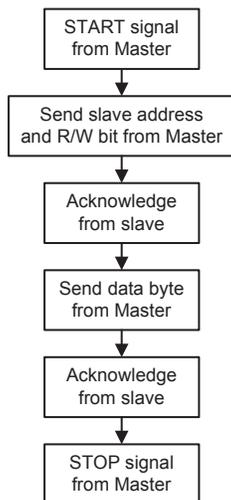
### **I<sup>2</sup>C Interface Operation**

The I<sup>2</sup>C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I<sup>2</sup>C bus is identified by a unique address which will be transmitted and received on the I<sup>2</sup>C bus.

When two devices communicate with each other on the bidirectional I<sup>2</sup>C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For the device, which only operates in slave mode, there are two methods of transferring data on the I<sup>2</sup>C bus, the slave transmit mode and the slave receive mode. The pull-high control function pin-shared with SCL/SDA pin is still applicable even if I<sup>2</sup>C device is activated and the related internal pull-high register could be controlled by its corresponding pull-high control register.



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



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

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

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

I<sup>2</sup>C Minimum  $f_{SYS}$  Frequency Requirements

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

There are three control registers associated with the I<sup>2</sup>C bus, SIMC0, SIMC1 and SIMTOC, one address register SIMA and one data register, SIMD. Note that the SIMC1, SIMD, SIMA and SIMTOC registers and their POR values are only available when the I<sup>2</sup>C mode is selected by properly configuring the UMD and SIM2~SIM0 bits in the SIMC0 register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

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

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

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I<sup>2</sup>C functions. Before the device writes data to the I<sup>2</sup>C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I<sup>2</sup>C bus, the device can read it from the SIMD register. Any transmission or reception of data from the I<sup>2</sup>C bus must be made via the SIMD register.

#### • SIMD Register

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

"x": Unknown

Bit 7~0      **D7~D0:** USIM SPI/I<sup>2</sup>C data register bit 7~bit 0

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

The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the SIMA register define the device slave address. Bit 0 is not defined. When a master device, which is connected to the I<sup>2</sup>C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected.

#### • SIMA Register

Bit	7	6	5	4	3	2	1	0
Name	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~1      **SIMA6~SIMA0:** I<sup>2</sup>C slave address  
 SIMA6~SIMA0 is the 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 SIMC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC1 register contains the relevant flags which are used to indicate the I<sup>2</sup>C communication status. Another register, SIMTOC, is used to control the I<sup>2</sup>C time-out function and is described in the corresponding section.

#### • SIMC0 Register

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I<sup>2</sup>C Operating Mode Control  
 000: SPI master mode; SPI clock is  $f_{SYS}/4$   
 001: SPI master mode; SPI clock is  $f_{SYS}/16$   
 010: SPI master mode; SPI clock is  $f_{SYS}/64$   
 011: SPI master mode; SPI clock is  $f_{SUB}$   
 100: SPI master mode; SPI clock is STM0 CCRP match frequency/2  
 101: SPI slave mode  
 110: I<sup>2</sup>C slave mode  
 111: Unused mode

When the UMD bit is cleared to zero, these bits setup the SPI or I<sup>2</sup>C operating mode of the USIM function. As well as selecting if the I<sup>2</sup>C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from STM0 and  $f_{SUB}$ . If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UMD**: UART mode selection bit  
 0: SPI or I<sup>2</sup>C mode  
 1: UART mode

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

Bit 3~2 **SIMDEB1~SIMDEB0**: I<sup>2</sup>C Debounce Time Selection  
 00: No debounce  
 01: 2 system clock debounce  
 1x: 4 system clock debounce

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

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

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

at the previous settings and should therefore be first initialised by the application program while the relevant I<sup>2</sup>C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

**Bit 0**     **SIMICF:** USIM SPI Incomplete Flag  
 This bit is only available when the USIM is configured to operate in an SPI slave mode. Refer to the SPI register section.

• **SIMC1 Register**

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

**Bit 7**     **HCF:** I<sup>2</sup>C Bus data transfer completion flag  
 0: Data is being transferred  
 1: Completion of an 8-bit data transfer  
 The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

**Bit 6**     **HAAS:** I<sup>2</sup>C Bus address match flag  
 0: Not address match  
 1: Address match  
 The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

**Bit 5**     **HBB:** I<sup>2</sup>C Bus busy flag  
 0: I<sup>2</sup>C Bus is not busy  
 1: I<sup>2</sup>C Bus is busy  
 The HBB flag is the I<sup>2</sup>C busy flag. This flag will be “1” when the I<sup>2</sup>C bus is busy which will occur when a START signal is detected. The flag will be set to “0” when the bus is free which will occur when a STOP signal is detected.

**Bit 4**     **HTX:** I<sup>2</sup>C slave device is transmitter or receiver selection  
 0: Slave device is the receiver  
 1: Slave device is the transmitter

**Bit 3**     **TXAK:** I<sup>2</sup>C Bus transmit acknowledge flag  
 0: Slave send acknowledge flag  
 1: Slave do not send acknowledge flag  
 The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8 bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set TXAK bit to “0” before further data is received.

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

**Bit 1**     **IAMWU:** I<sup>2</sup>C Address Match Wake-up control  
 0: Disable  
 1: Enable

This bit should be set 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.

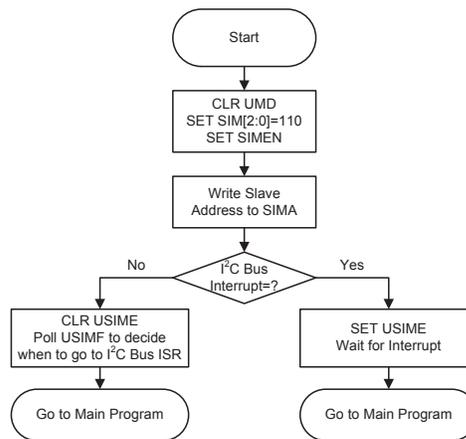
Bit 0 **RXAK:** I<sup>2</sup>C Bus Receive acknowledge flag  
 0: Slave receive acknowledge flag  
 1: Slave does not receive acknowledge flag

The RXAK flag is the receiver acknowledge flag. When the RXAK flag is “0”, it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is “1”. When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I<sup>2</sup>C Bus.

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

Communication on the I<sup>2</sup>C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I<sup>2</sup>C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the SIMC1 register will be set and a USIM interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer completion or from the I<sup>2</sup>C bus time-out occurrence. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I<sup>2</sup>C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

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



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

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

The START signal can only be generated by the master device connected to the I<sup>2</sup>C bus and not by the slave device. This START signal will be detected by all devices connected to the I<sup>2</sup>C bus. When detected, this indicates that the I<sup>2</sup>C bus is busy and therefore the HBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.

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

The transmission of a START signal by the master will be detected by all devices on the I<sup>2</sup>C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal USIM I<sup>2</sup>C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the SIMC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

As an USIM I<sup>2</sup>C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer or from the I<sup>2</sup>C bus time-out occurrence. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

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

The SRW bit in the SIMC1 register defines whether the master device wishes to read data from the I<sup>2</sup>C bus or write data to the I<sup>2</sup>C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is “1” then this indicates that the master device wishes to read data from the I<sup>2</sup>C bus, therefore the slave device must be setup to send data to the I<sup>2</sup>C bus as a transmitter. If the SRW flag is “0” then this indicates that the master wishes to send data to the I<sup>2</sup>C bus, therefore the slave device must be setup to read data from the I<sup>2</sup>C bus as a receiver.

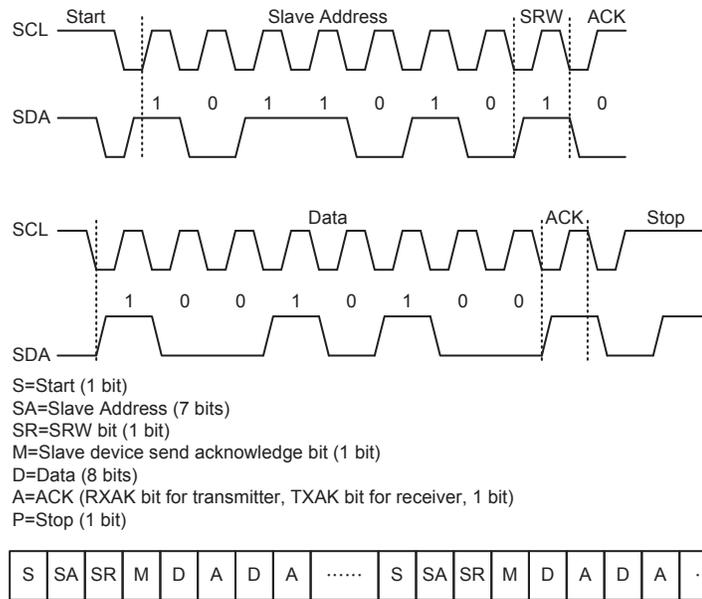
### **I<sup>2</sup>C Bus Slave Address Acknowledge Signal**

After the master has transmitted a calling address, any slave device on the I<sup>2</sup>C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to “1”. If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be set to “0”.

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

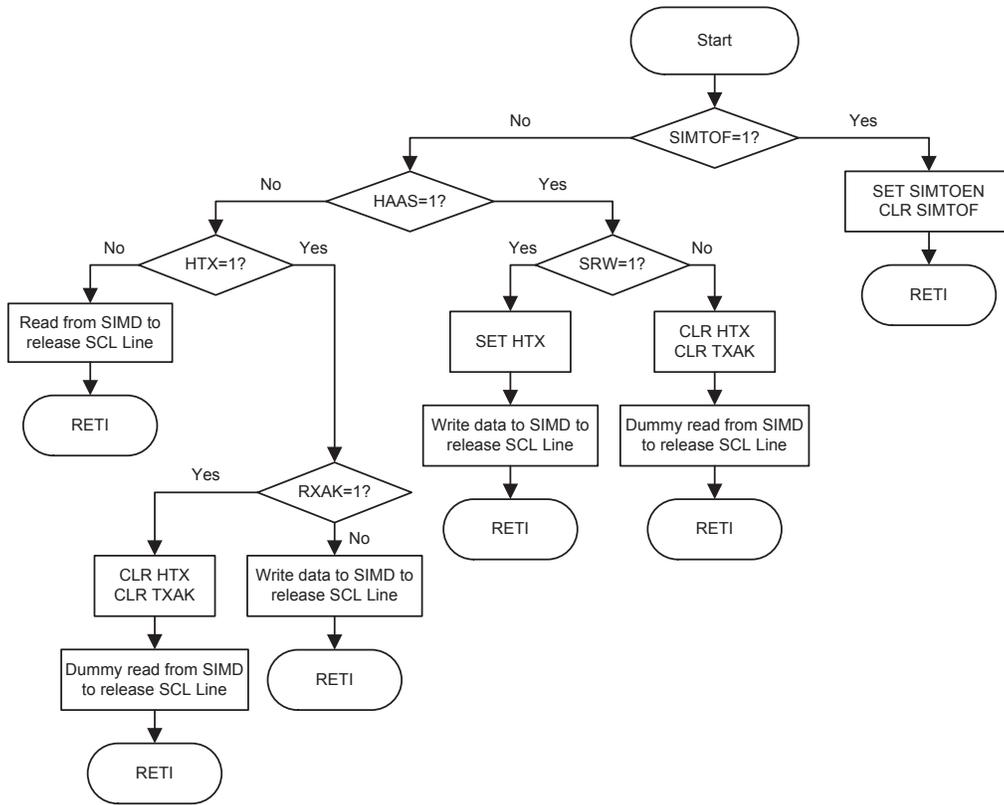
The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level “0”, before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I<sup>2</sup>C Bus. The corresponding data will be stored in the SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register.

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



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

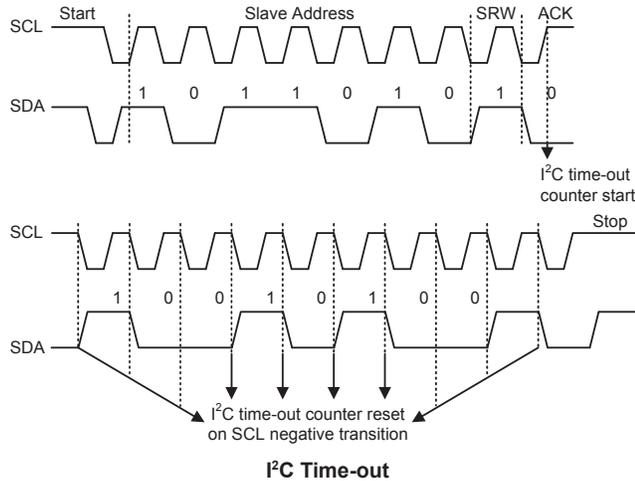
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.



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

**I<sup>2</sup>C Time-out Control**

In order to reduce the problem of I<sup>2</sup>C lockup due to reception of erroneous clock sources, a time-out function is provided. If the clock source to the I<sup>2</sup>C is not received for a while, then the I<sup>2</sup>C circuitry and registers will be reset after a certain time-out period. The time-out counter starts counting on an I<sup>2</sup>C bus “START” & “address match” condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out setup by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I<sup>2</sup>C “STOP” condition occurs.



When an I<sup>2</sup>C time-out counter overflow occurs, the counter will stop and the SIMTOEN bit will be cleared to zero and the SIMTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the USIM interrupt vector. When an I<sup>2</sup>C time-out occurs, the I<sup>2</sup>C internal circuitry will be reset and the registers will be reset into the following condition:

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

**I<sup>2</sup>C Registers after Time-out**

The SIMTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using SIMTOS bit field in the SIMTOC register. The time-out time is given by the formula:  $((1\sim64)\times32)/f_{SUB}$ . This gives a time-out period which ranges from about 1ms to 64ms.

• **SIMTOC Register**

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

Bit 7      **SIMTOEN**: USIM I<sup>2</sup>C Time-out control  
             0: Disable  
             1: Enable

Bit 6      **SIMTOF**: USIM I<sup>2</sup>C Time-out flag  
             0: No time-out occurred  
             1: Time-out occurred

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

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

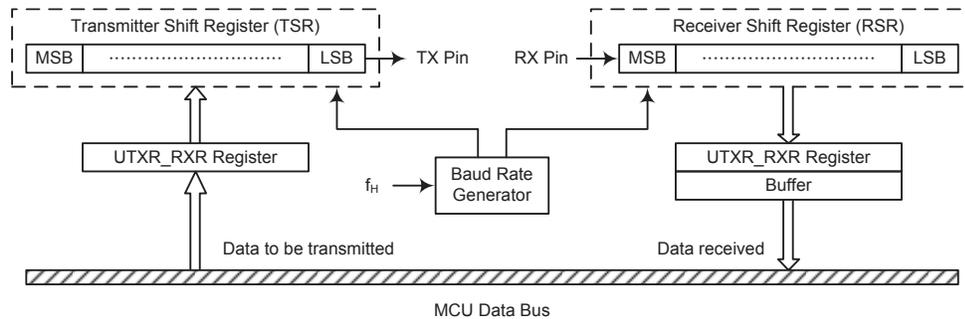
**UART Interface**

The device contains an integrated full-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function shares the same internal interrupt vector with the SPI and I<sup>2</sup>C interfaces which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UART function contains the following features:

- Full-duplex, asynchronous communication
- 8 or 9 bits character length
- Even, odd or no parity options
- One or two stop bits
- Baud rate generator with 8-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver

- 2-byte Deep FIFO Receive Data Buffer
- RX pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
  - ♦ Transmitter Empty
  - ♦ Transmitter Idle
  - ♦ Receiver Full
  - ♦ Receiver Overrun
  - ♦ Address Mode Detect



**UART Data Transfer Block Diagram**

### UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as TX and RX. The TX and RX pins are the UART transmitter and receiver pins respectively. The TX and RX pin function should first be selected by the corresponding pin-shared function selection register before the UART function is used. Along with the UMD bit, the UREN bit, the UTXEN and URXEN bits, if set, will setup these pins to their respective TX output and RX input conditions and disable any pull-high resistor option which may exist on the TX and RX pins. When the TX or RX pin function is disabled by clearing the UMD, UREN, UTXEN or URXEN bit, the TX or RX pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the TX or RX pin or not is determined by the corresponding I/O pull-high function control bit.

### UART Data Transfer Scheme

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

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

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

### UART Status and Control Registers

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

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

**UART Register List**

#### • SIMC0 Register

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

- Bit 7~5     **SIM2~SIM0**: USIM SPI/I<sup>2</sup>C Operating Mode Control  
When the UMD bit is cleared to zero, these bits setup the SPI or I<sup>2</sup>C operating mode of the USIM function. Refer to the SPI or I<sup>2</sup>C register section for more details.
- Bit 4        **UMD**: UART mode selection bit  
0: SPI or I<sup>2</sup>C mode  
1: UART mode  
This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I<sup>2</sup>C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I<sup>2</sup>C mode.
- Bit 3~2     **SIMDEB1~SIMDEB0**: I<sup>2</sup>C Debounce Time Selection  
Refer to the I<sup>2</sup>C register section.
- Bit 1        **SIMEN**: USIM SPI/I<sup>2</sup>C Enable Control  
This bit is only available when the USIM is configured to operate in an SPI or I<sup>2</sup>C mode with the UMD bit set low. Refer to the SPI or I<sup>2</sup>C register section for more details.
- Bit 0        **SIMICF**: USIM SPI Incomplete Flag  
Refer to the SPI register section.

• **UUSR Register**

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

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

**Bit 7**      **UPERR:** Parity error flag  
                  0: No parity error is detected  
                  1: Parity error is detected  
 The UPERR flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if Parity mode (odd or even) is selected. The flag can also be cleared by a software sequence which involves a read to the status register UUSR followed by an access to the UTXR\_RXR data register.

**Bit 6**      **UNF:** Noise flag  
                  0: No noise is detected  
                  1: Noise is detected  
 The UNF flag is the noise flag. When this read only flag is “0”, it indicates no noise condition. When the flag is “1”, it indicates that the UART has detected noise on the receiver input. The UNF flag is set during the same cycle as the URXIF flag but will not be set in the case of an overrun. The UNF flag can be cleared by a software sequence which will involve a read to the status register UUSR followed by an access to the UTXR\_RXR data register.

**Bit 5**      **UFERR:** Framing error flag  
                  0: No framing error is detected  
                  1: Framing error is detected  
 The UFERR flag is the framing error flag. When this read only flag is “0”, it indicates that there is no framing error. When the flag is “1”, it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register UUSR followed by an access to the UTXR\_RXR data register.

**Bit 4**      **UOERR:** Overrun error flag  
                  0: No overrun error is detected  
                  1: Overrun error is detected  
 The UOERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is “0”, it indicates that there is no overrun error. When the flag is “1”, it indicates that an overrun error occurs which will inhibit further transfers to the UTXR\_RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register UUSR followed by an access to the UTXR\_RXR data register.

**Bit 3**      **URIDLE:** Receiver status  
                  0: Data reception is in progress (Data being received)  
                  1: No data reception is in progress (Receiver is idle)  
 The URIDLE flag is the receiver status flag. When this read only flag is “0”, it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is “1”, it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the URIDLE bit is “1” indicating that the UART receiver is idle and the RX pin stays in logic high condition.

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

• **UUCR1 Register**

The UUCR1 register together with the UUCR2 register are the two UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

“x”: Unknown

- Bit 7     **UREN**: UART function enable control  
           0: Disable UART. TX and RX pins are in a floating state  
           1: Enable UART. TX and RX pins function as UART pins  
 The UREN bit is the UART enable bit. When this bit is equal to “0”, the UART will be disabled and the RX pin as well as the TX pin will be set in a floating state. When the bit is equal to “1”, the UART will be enabled if the UMD bit is set and the TX and RX pins will function as defined by the UTXEN and URXEN enable control bits.  
 When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF bits will

be cleared, while the UTIDLE, UTXIF and URIDLE bits will be set. Other control bits in UUCR1, UUCR2 and UBRG registers will remain unaffected. If the UART is active and the UREN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

- Bit 6     **UBNO**: Number of data transfer bits selection  
          0: 8-bit data transfer  
          1: 9-bit data transfer
- This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to “1”, a 9-bit data length format will be selected. If the bit is equal to “0”, then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits URX8 and UTX8 will be used to store the 9th bit of the received and transmitted data respectively.
- Bit 5     **UPREN**: Parity function enable control  
          0: Parity function is disabled  
          1: Parity function is enabled
- This is the parity enable bit. When this bit is equal to “1”, the parity function will be enabled. If the bit is equal to “0”, then the parity function will be disabled.
- Bit 4     **UPRT**: Parity type selection bit  
          0: Even parity for parity generator  
          1: Odd parity for parity generator
- This bit is the parity type selection bit. When this bit is equal to “1”, odd parity type will be selected. If the bit is equal to “0”, then even parity type will be selected.
- Bit 3     **USTOPS**: Number of Stop bits selection  
          0: One stop bit format is used  
          1: Two stop bits format is used
- This bit determines if one or two stop bits are to be used. When this bit is equal to “1”, two stop bits are used. If this bit is equal to “0”, then only one stop bit is used.
- Bit 2     **UTXBRK**: Transmit break character  
          0: No break character is transmitted  
          1: Break characters transmit
- The UTXBRK bit is the Transmit Break Character bit. When this bit is “0”, there are no break characters and the TX pin operates normally. When the bit is “1”, there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to “1”, after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the UTXBRK bit is reset.
- Bit 1     **URX8**: Receive data bit 8 for 9-bit data transfer format (read only)
- This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as URX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.
- Bit 0     **UTX8**: Transmit data bit 8 for 9-bit data transfer format (write only)
- This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as UTX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UUCR2 Register**

The UUCR2 register is the second of the two UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UART Transmitter and Receiver as well as enabling the various USIM UART mode interrupt sources. The register also serves to control the baud rate speed, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

**Bit 7 UTXEN: UART Transmitter enabled control**

- 0: UART transmitter is disabled
- 1: UART transmitter is enabled

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

If the UTXEN bit is equal to “1” and the UMD and UREN bit are also equal to “1”, the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the UTXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be set in a floating state.

**Bit 6 URXEN: UART Receiver enabled control**

- 0: UART receiver is disabled
- 1: UART receiver is enabled

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

**Bit 5 UBRGH: Baud Rate speed selection**

- 0: Low speed baud rate
- 1: High speed baud rate

The bit named UBRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register UBRG, controls the Baud Rate of the UART. If this bit is equal to “1”, the high speed mode is selected. If the bit is equal to “0”, the low speed mode is selected.

**Bit 4 UADDEN: Address detect function enable control**

- 0: Address detect function is disabled
- 1: Address detect function is enabled

The bit named UADDEN is the address detect function enable control bit. When this bit is equal to “1”, the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to URX7 if UBNO=0 or the 9th bit, which corresponds to URX8 if UBNO=1, has a value of “1”, then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of UBNO. If the address bit known as the 8th or 9th bit of the received word is “0” with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

- Bit 3      **UWAKE:** RX pin wake-up UART function enable control  
             0: RX pin wake-up UART function is disabled  
             1: RX pin wake-up UART function is enabled  
 This bit is used to control the wake-up UART function when a falling edge on the RX pin occurs. Note that this bit is only available when the UART clock ( $f_{H1}$ ) is switched off. There will be no RX pin wake-up UART function if the UART clock ( $f_{H1}$ ) exists. If the UWAKE bit is set to 1 as the UART clock ( $f_{H1}$ ) is switched off, a UART wake-up request will be initiated when a falling edge on the RX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock ( $f_{H1}$ ) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the RX pin when the UWAKE bit is cleared to 0.
- Bit 2      **URIE:** Receiver interrupt enable control  
             0: Receiver related interrupt is disabled  
             1: Receiver related interrupt is enabled  
 This bit enables or disables the receiver interrupt. If this bit is equal to “1” and when the receiver overrun flag UOERR or receive data available flag URXIF is set, the USIM interrupt request flag USIMF will be set. If this bit is equal to “0”, the USIM interrupt request flag USIMF will not be influenced by the condition of the UOERR or URXIF flags.
- Bit 1      **UTIE:** Transmitter Idle interrupt enable control  
             0: Transmitter idle interrupt is disabled  
             1: Transmitter idle interrupt is enabled  
 This bit enables or disables the transmitter idle interrupt. If this bit is equal to “1” and when the transmitter idle flag UTIDLE is set, due to a transmitter idle condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to “0”, the USIM interrupt request flag USIMF will not be influenced by the condition of the UTIDLE flag.
- Bit 0      **UTEIE:** Transmitter Empty interrupt enable control  
             0: Transmitter empty interrupt is disabled  
             1: Transmitter empty interrupt is enabled  
 This bit enables or disables the transmitter empty interrupt. If this bit is equal to “1” and when the transmitter empty flag UTXIF is set, due to a transmitter empty condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to “0”, the USIM interrupt request flag USIMF will not be influenced by the condition of the UTXIF flag.

• **UTXR\_RXR Register**

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

Bit	7	6	5	4	3	2	1	0
Name	UTXR7	UTXR6	UTXR5	UTXR4	UTXR3	UTXR2	UTXR1	UTXR0
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      **UTXR7~UTXR0:** UART Transmit/Receive Data bit 7~bit 0

• **UBRG Register**

Bit	7	6	5	4	3	2	1	0
Name	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”: Unknown

Bit 7~0     **UBRG7~UBRG0:** Baud Rate values

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

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

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

**Baud Rate Generator**

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the baud rate register UBRG and the second is the value of the UBRGH bit with the control register UUCR2. The UBRGH bit decides if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value N in the UBRG register which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the UBRG register and has a range of between 0 and 255.

UUCR2 UBRGH Bit	0	1
Baud Rate (BR)	$f_H/[64 (N+1)]$	$f_H/[16 (N+1)]$

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

**Calculating the Baud Rate and Error Values**

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

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

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

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

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

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

**UART Setup and Control**

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding UBNO, UPRT, UPREN, and USTOPS

bits in the UUCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

**Enabling/Disabling the UART Interface**

The basic on/off function of the internal UART function is controlled using the UREN bit in the UUCR1 register. When the UART mode is selected by setting the UMD bit in the SIMC0 register to “1”, if the UREN, UTXEN and URXEN bits are set, then these two UART pins will act as normal TX output pin and RX input pin respectively. If no data is being transmitted on the TX pin, then it will default to a logic high value.

Clearing the UREN bit will disable the TX and RX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF being cleared while bits UTIDLE, UTXIF and URIDLE will be set. The remaining control bits in the UUCR1, UUCR2 and UBRG registers will remain unaffected. If the UREN bit in the UUCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

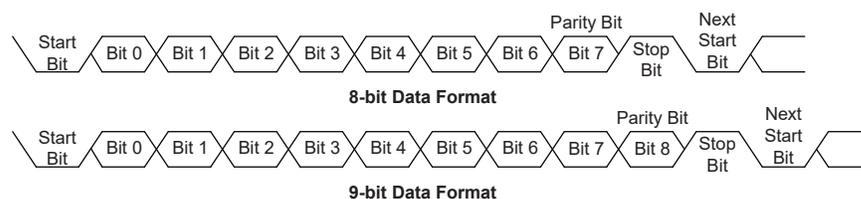
**Data, Parity and Stop Bit Selection**

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

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

**Transmitter Receiver Data Format**

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



### UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the UBNO bit in the UUCR1 register. When UBNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the UTX8 bit in the UUCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the UTXR\_RXR register. The data to be transmitted is loaded into this UTXR\_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the UTXR\_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the UTXEN bit is set, but the data will not be transmitted until the UTXR\_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the UTXR\_RXR register, after which the UTXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the UTXR\_RXR register will result in an immediate transfer to the TSR. If during a transmission the UTXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

### Transmitting Data

When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant bit first. In the transmit mode, the UTXR\_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the UTX8 bit in the UUCR1 register. The steps to initiate a data transfer can be summarized as follows:

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

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

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

1. A UUSR register access
2. A UTXR\_RXR register write execution

The read-only UTXIF flag is set by the UART hardware and if set indicates that the UTXR\_RXR register is empty and that other data can now be written into the UTXR\_RXR register without overwriting the previous data. If the UTEIE bit is set then the UTXIF flag will generate an interrupt. During a data transmission, a write instruction to the UTXR\_RXR register will place the data into the UTXR\_RXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the UTXR\_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the UTXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the UTIDLE bit will be set. To clear the UTIDLE bit the following software sequence is used:

1. A UUSR register access
2. A UTXR\_RXR register write execution

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

### **Transmit Break**

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

### **UART Receiver**

The UART is capable of receiving word lengths of either 8 or 9 bits. If the UBNO bit is set, the word length will be set to 9 bits with the MSB being stored in the URX8 bit of the UUCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

### **Receiving Data**

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

- Make the correct selection of UBNO, UPRT and UPREN bits to define the word length, parity type.
- Setup the UBRG register to select the desired baud rate.
- Set the URXEN bit to ensure that the RX pin is used as a UART receiver pin.

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

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

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

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

1. A UUSR register access
2. A UTXR\_RXR register read execution

### **Receive Break**

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

- The framing error flag, UFERR, will be set.
- The receive data register, UTXR\_RXR, will be cleared.
- The UOERR, UNF, UPERR, URIDLE or URXIF flags will possibly be set.

### **Idle Status**

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

### **Receiver Interrupt**

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

## **Managing Receiver Errors**

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

### **Overrun Error – UOERR**

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

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

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

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

### **Noise Error – UNF**

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

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

Note that the UNF flag is reset by a UUSR register read operation followed by a UTXR\_RXR register read operation.

### **Framing Error – UFERR**

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

### **Parity Error – UPERR**

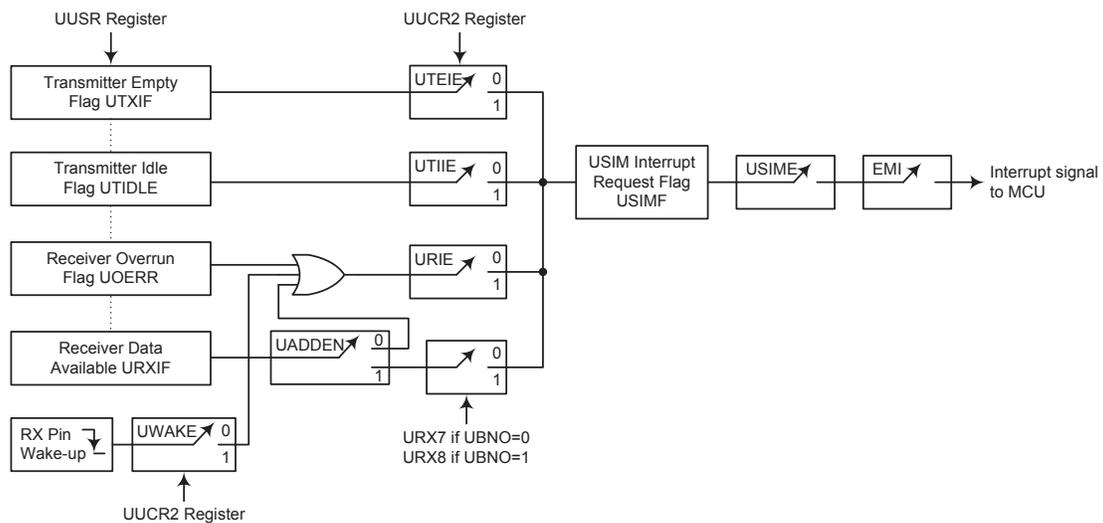
The read only parity error flag, UPERR, in the UUSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, UPREN=1, and if the parity type, odd or even is selected. The read only UPERR flag and the received data will be recorded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

### UART Interrupt Structure

Several individual UART conditions can trigger an USIM interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and the USIM interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding UUSR register flags which will generate an USIM interrupt if its associated interrupt enable control bit in the UUCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual USIM UART mode interrupt sources.

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

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



UART Interrupt Structure

**Address Detect Mode**

Setting the Address Detect Mode bit, UADDEN, in the UUCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the URXIF flag. If the UADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the USIME and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if UBNO=1 or the 8th bit if UBNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the UADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the URXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit UPREN to zero.

UADDEN	Bit 9 if UBNO=1, Bit 8 if UBNO=0	USIM Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

**UADDEN Bit Function**

**UART Power Down and Wake-up**

When the UART clock ( $f_{H}$ ) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock ( $f_{H}$ ) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the IDLE or SLEEP Mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP Mode, note that the UUSR, UUCR1, UUCR2, transmit and receive registers, as well as the UBRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the UWAKE bit in the UUCR2 register. If this bit, along with the UART mode selection bit, UMD, the UART enable bit, UREN, the receiver enable bit, URXEN and the receiver interrupt bit, URIE, are all set when the UART clock ( $f_{H}$ ) is off, then a falling edge on the RX pin will trigger an RX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX pin will be ignored.

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

## LDO Function

The device contains a low power voltage regulator implemented in CMOS technology. Using CMOS technology ensures low voltage drop and low quiescent current. The output voltages can range from 2.2V~3.6V, which can be setup by the bits, VSEL3~VSEL0, in the LDOC register.

### • LDOC Register

Bit	7	6	5	4	3	2	1	0
Name	LDOEN	LDOM	VREGS	VSW	VSEL3	VSEL2	VSEL1	VSEL0
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

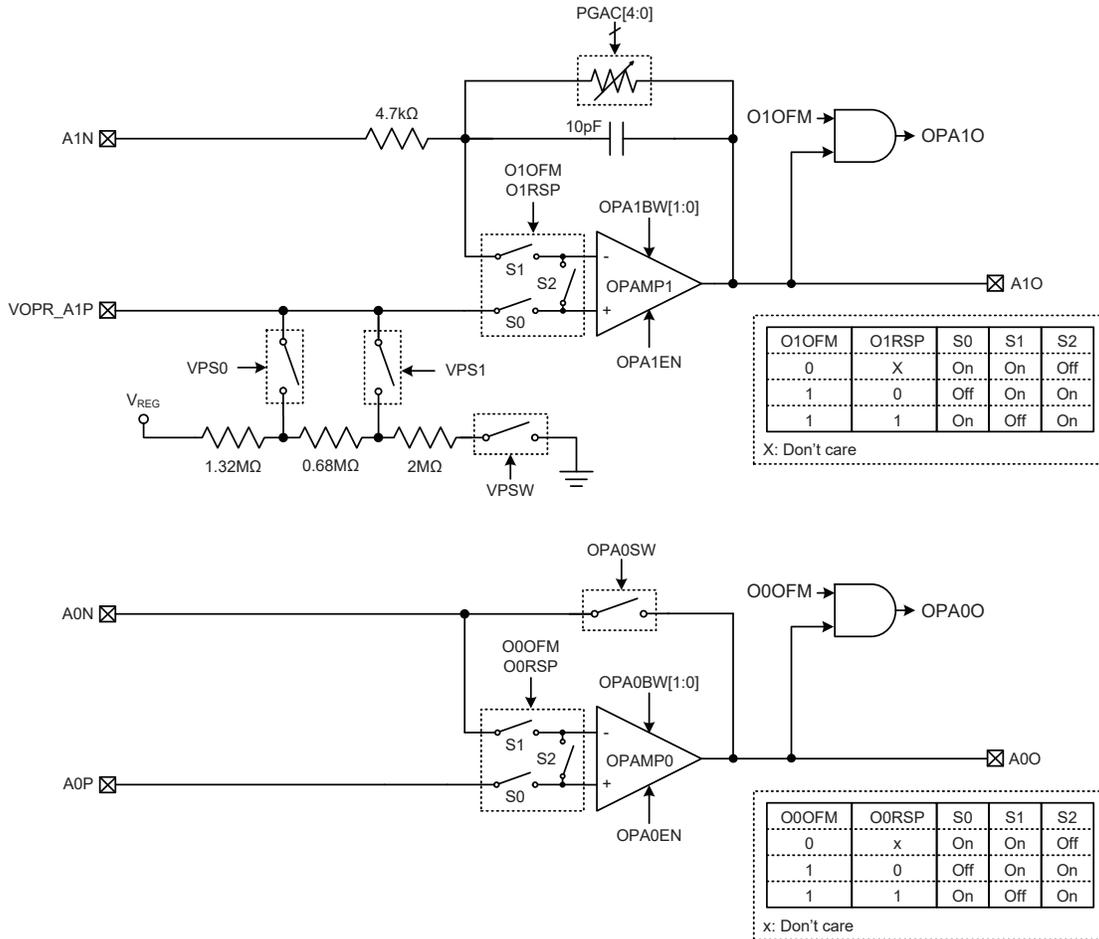
- Bit 7      **LDOEN**: LDO control bit  
             0: Disable  
             1: Enable
- Bit 6      **LDOM**: LDO power mode selection  
             0: Power-saving mode (2/3 bias current)  
             1: Normal mode
- Bit 5      **VREGS**: VREG status when LDOEN=0 and VSW=0  
             0: Weak pull low  
             1: Floating
- Bit 4      **VSW**: VREG voltage source selection  
             0: LDO  
             1:  $V_{DD}$
- Bit 3~0   **VSEL3~VSEL0**: LDO output voltage selection (VPSW=1)  
             0000: 2.2V  
             0001: 2.3V  
             0010: 2.4V  
             0011: 2.5V  
             0100: 2.6V  
             0101: 2.7V  
             0110: 2.8V  
             0111: 2.9V  
             1000: 3.0V  
             1001: 3.1V  
             1010: 3.2V  
             1011: 3.3V  
             1100: 3.4V  
             1101: 3.5V  
             1110: 3.6V  
             1111: 3.6V

Note: When the bit VPSW is 0, the LDO output voltage will not be selected by these four bits, and the voltage will be about 1.25V.

- Note: 1. When the  $V_{DD}$  voltage is stable and after switching on the LDO, the LDO output voltage will be stable after 20 $\mu$ s.  
 2. When users change the LDO output voltage, the LDO output voltage will be stable after 12ms.  
 3. If the LDO output is as the ADC reference voltage, then the VCAP should be a 0.1 $\mu$ F capacitor connected to ground.  
 4. The LDO input voltage ( $V_{DD}$ ) must be 0.1V greater than output voltage for obtaining stable output voltage.

## Operational Amplifiers

There are two Operational Amplifiers in the device, OPA0 and OPA1. The OPA0 amplifier is setup for a band pass filter application circuit. For PIR applications, it is recommended that the bandwidth is setup from 0.3 to 8Hz. The OPA1 amplifier is a programmable gain amplifier whose gain can be setup to have a range of 128 to 376. This gain is setup using the application software.



**Operational Amplifiers Block Diagram**

### Operational Amplifier Registers

The Operational Amplifiers are fully under the control of several internal registers. OPAC0 and OPAC1. These registers control enable/disable function.

#### • OPAC0 Register

Bit	7	6	5	4	3	2	1	0
Name	OPA1EN	OPA0EN	VPSW	—	—	OPA0SW	VPS1	VPS0
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

- Bit 7      **OPA1EN**: OPA1 enable or disable control bit  
           0: Disable  
           1: Enable
- Bit 6      **OPA0EN**: OPA0 enable or disable control bit  
           0: Disable  
           1: Enable
- Bit 5      **VPSW**: Switch control bit  
           0: Off  
           1: On
- Bit 4~3    Unimplemented, read as “0”
- Bit 2      **OPA0SW**: OPA0 short switch control  
           0: Off  
           1: On
- Bit 1      **VPS1**:  $V_P$  voltage selection  
           0: Floating  
           1:  $1/2V_{REG}$  when VPSW is set to 1
- Bit 0      **VPS0**:  $V_P$  voltage selection  
           0: Floating  
           1:  $2/3V_{REG}$  when VPSW is set to 1
- Note: VPS0 bit and VPS1 bit can not be 1 at the same time.

#### • OPAC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	PGAC4	PGAC3	PGAC2	PGAC1	PGAC0
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    **PGAC4~PGAC0**: OPA1 gain control bit  
           Gain= $128+(PGAC \times 8)$

• **OPAC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	OPA1O	—	OPA1BW1	OPA1BW0	OPA0O	—	OPA0BW1	OPA0BW0
R/W	R	—	R/W	R/W	R	—	R/W	R/W
POR	0	—	0	0	0	—	0	0

- Bit 7~5     **OPA1O**: OPA1 digital output bit; positive logic (read only)  
The OPA1O is “0” when the OPA1 is disabled.  
When O1OFM is set to 1, OPA1O is defined as OPA1 output status.  
Note: For the OPA1, when executing the offset calibration function, the A1N state must be floating to ensure the value accuracy. It is not recommended to use A1N for offset calibration as there are internal components on this port.
- Bit 6       Unimplemented, read as “0”
- Bit 5~4     **OPA1BW1~OPA1BW0**: OPA1 bandwidth control bits (refer to OP Amplifier Electrical Characteristics)  
00: 5kHz  
01: 40kHz  
10: 600kHz  
11: 2MHz
- Bit 3       **OPA0O**: OPA0 digital output bit; positive logic (read only)  
The OPA0O is “0” when the OPA0 is disabled.  
When O0OFM is set to 1, OPA0O is defined as OPA0 output status.
- Bit 2       Unimplemented, read as “0”
- Bit 1~0     **OPA0BW1~OPA0BW0**: OPA0 bandwidth control bits (refer to OP Amplifier Electrical Characteristics)  
00: 5kHz  
01: 40kHz  
10: 600kHz  
11: 2MHz

• **OPA0VOS Register**

Bit	7	6	5	4	3	2	1	0
Name	O0OFM	O0RSP	O0OF5	O0OF4	O0OF3	O0OF2	O0OF1	O0OF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

- Bit 7       **O0OFM**: OPA0 normal operation or input offset voltage calibration mode selection bit  
0: Normal operation  
1: Offset calibration mode
- Bit 6       **O0RSP**: OPA0 input offset voltage calibration reference selection bit  
0: Input reference voltage comes from A0N  
1: Input reference voltage comes from A0P
- Bit 5~0     **O0OF5 ~ O0OF0**: OPA0 input offset voltage calibration control bits



## 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 one external interrupt and several internal interrupts functions. The external interrupt is generated by the action of the external INTn pins, while the internal interrupts are generated by various internal functions such as Timer Modules, Time Bases, LVD, EEPROM, USIM and the A/D converter, etc.

### Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The registers fall into three categories. The first is the INTC0~INTC1 registers which setup the primary interrupts, the second is the MFInC 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=0~1
Auto Conversion Function	ACFE	ACFF	—
Time Bases	TBnF	TBnE	n=0~1
Multi-function	MFnE	MFnF	n=0~1
STM	STMnAE	STMnAF	n=0~1
	STMnPE	STMnPF	
A/D Converter	ADE	ADF	—
USIM	USIME	USIMF	—
EEPROM	DEE	DEF	—
LVD	LVE	LVF	—

**Interrupt Register Bit Naming Conventions**

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	ACFF	INT1F	INT0F	ACFE	INT1E	INT0E	EMI
INTC1	MF1F	MF0F	TB1F	TB0F	MF1E	MF0E	TB1E	TB0E
MFIOC	STM1AF	STM1PF	STM0AF	STM0PF	STM1AE	STM1PE	STM0AE	STM0PE
MF11C	LVF	DEF	USIMF	ADF	LVE	DEE	USIME	ADE

**Interrupt Register List**

• **INTEG Register**

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

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

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	ACFF	INT1F	INT0F	ACFE	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 **ACFF**: Auto Conversion Function 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 **ACFE**: Auto Conversion Function 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	MF1F	MF0F	TB1F	TB0F	MF1E	MF0E	TB1E	TB0E
R/W								
POR	0	0	0	0	0	0	0	0

- Bit 7      **MF1F**: Multi-function Interrupt 1 request flag  
0: No request  
1: Interrupt request
- Bit 6      **MF0F**: Multi-function Interrupt 0 request flag  
0: No request  
1: Interrupt request
- Bit 5      **TB1F**: Time Base 1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **TB0F**: Time Base 0 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3      **MF1E**: Multi-function interrupt 1 control  
0: Disable  
1: Enable
- Bit 2      **MF0E**: Multi-function interrupt 0 control  
0: Disable  
1: Enable
- Bit 1      **TB1E**: Time Base 1 interrupt control  
0: Disable  
1: Enable
- Bit 0      **TB0E**: Time Base 0 interrupt control  
0: Disable  
1: Enable

• **MFIOC Register**

Bit	7	6	5	4	3	2	1	0
Name	STM1AF	STM1PF	STM0AF	STM0PF	STM1AE	STM1PE	STM0AE	STM0PE
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      **STM1AF**: STM1 Comparator A match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6      **STM1PF**: STM1 Comparator P match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **STM0AF**: STM0 Comparator A match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **STM0PF**: STM0 Comparator P match interrupt request flag  
0: No request  
1: Interrupt request

- Bit 3      **STM1AE**: STM1 Comparator A match interrupt control  
0: Disable  
1: Enable
- Bit 2      **STM1PE**: STM1 Comparator P match interrupt control  
0: Disable  
1: Enable
- Bit 1      **STM0AE**: STM0 Comparator A match interrupt control  
0: Disable  
1: Enable
- Bit 0      **STM0PE**: STM0 Comparator P match interrupt control  
0: Disable  
1: Enable

• **MF11C Register**

Bit	7	6	5	4	3	2	1	0
Name	LVF	DEF	USIMF	ADF	LVE	DEE	USIME	ADE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **LVF**: LVD interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6      **DEF**: Data EEPROM interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **USIMF**: USIM interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **ADF**: A/D converter interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3      **LVE**: LVD interrupt control  
0: Disable  
1: Enable
- Bit 2      **DEE**: Data EEPROM interrupt control  
0: Disable  
1: Enable
- Bit 1      **USIME**: USIM interrupt control  
0: Disable  
1: Enable
- Bit 0      **ADE**: A/D converter interrupt control  
0: Disable  
1: Enable

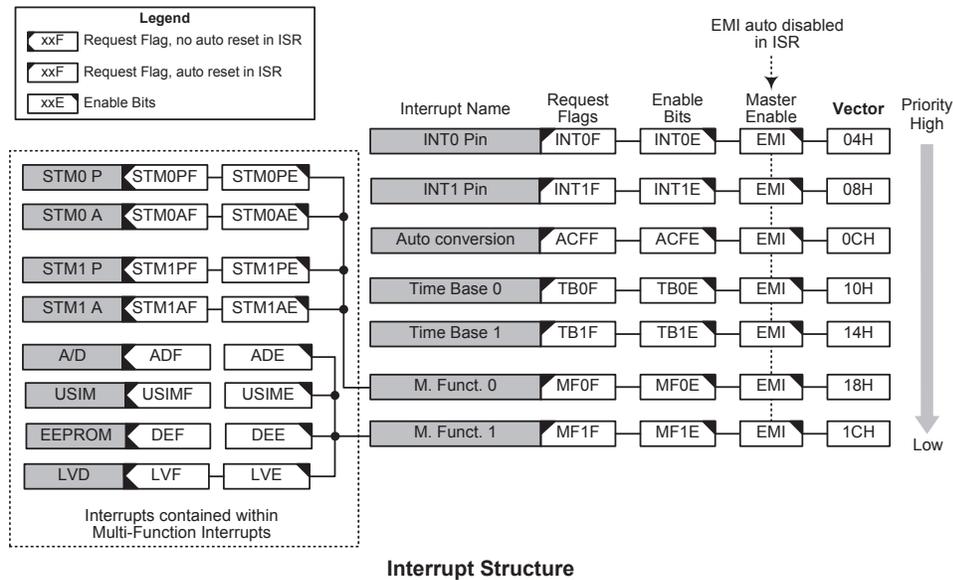
## Interrupt Operation

When the conditions for an interrupt event occur, such as a Timer Module compare 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 Interrupt

The external interrupt is controlled by signal transitions on the INTn pins. An external interrupt request will take place when the external interrupt request flags, INTnF, 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 bits, INTnE, must first be set. Additionally the correct interrupt edge type must be selected using the related 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. 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, INTnF, 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 pin will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

### Auto Conversion Function Interrupt

The device contains an Auto Conversion Function which has its own independent interrupt. An internal Auto Conversion Function interrupt will take place when the Auto Conversion Function interrupt request flag ACFF, is set, which occurs when the event counter value reaches the pre-programmed number of events. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Auto Conversion Function interrupt enable bit, ACFE, must first be set. When the interrupt is enabled, the stack is not full and the condition mentioned above occurs, a subroutine call to the Auto Conversion Function interrupt vector, will take place. When the interrupt is serviced, the Auto Conversion Function interrupt flag, ACFF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

## Timer Module Interrupts

The Standard type TMs each has two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. There are two interrupt request flags and two enable control bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MF0E, 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 MF0F 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.

## USIM Interrupt

The Universal Serial Interface Module Interrupt, also known as the USIM interrupt, which is contained within the multi-function interrupt, will take place when the USIM Interrupt request flag, USIMF, is set. As the USIM interface can operate in three modes which are SPI mode, I<sup>2</sup>C mode and UART mode, the USIMF flag can be set by different conditions depending on the selected interface mode.

If the SPI or I<sup>2</sup>C mode is selected, the USIM interrupt can be triggered when a byte of data has been received or transmitted by the USIM SPI or I<sup>2</sup>C interface, or an I<sup>2</sup>C slave address match occurs, or an I<sup>2</sup>C bus time-out occurs. If the UART mode is selected, several individual UART conditions including a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up, can generate a USIM interrupt with the USIMF flag bit set high.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, the Multi-function Interrupt enable bit, MF1E, and the Universal Serial Interface Module Interrupt enable bit, USIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the respective Interrupt vector, will take place. When the interrupt is serviced, The EMI bit will also be automatically cleared to disable other interrupts. However, only the related MF1F flag will be automatically cleared. As the USIM interrupt request flags will not be automatically cleared, it has to be cleared by the application program.

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

## Multi-function Interrupt

Within the 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, TM interrupts, A/D converter interrupt, LVD interrupt, USIM interrupt and EEPROM interrupt.

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

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

### **A/D Converter Interrupt**

The A/D converter interrupt is contained within the Multi-function 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 Interrupt enable bit, ADE, and associated Multi-function interrupt enable bit, 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 EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the ADF flag will not be automatically cleared, it has to be cleared by the application program.

### **EEPROM Interrupt**

The EEPROM interrupt is contained within the Multi-function Interrupt. An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the respective EEPROM Interrupt vector, will take place. When the EEPROM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

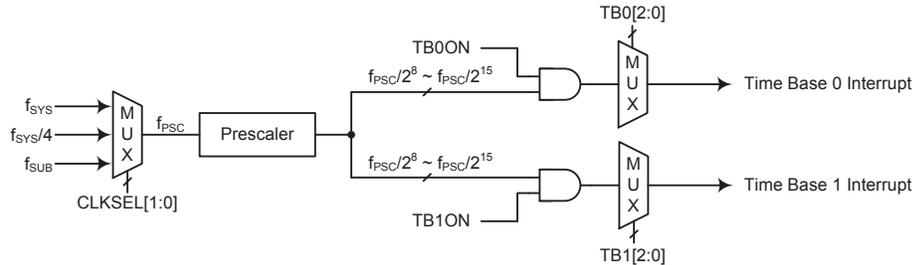
### **LVD Interrupt**

The Low Voltage Detector Interrupt is contained within the Multi-function Interrupt. An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the Multi-function Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the LVF flag will not be automatically cleared, it has to be cleared by the application program.

### **Time Base Interrupts**

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

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



**Time Base Interrupts**

• **PSCR Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source selection

- 00:  $f_{SYS}$
- 01:  $f_{SYS}/4$
- 1x:  $f_{SUB}$

• **TB0C Register**

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

Bit 7 **TB0ON**: Time Base 0 Control

- 0: Disable
- 1: Enable

Bit 6~3 Unimplemented, read as “0”

Bit 2~0 **TB02~TB00**: Select Time Base 0 Time-out Period

- 000:  $2^8/f_{PSC}$
- 001:  $2^9/f_{PSC}$
- 010:  $2^{10}/f_{PSC}$
- 011:  $2^{11}/f_{PSC}$
- 100:  $2^{12}/f_{PSC}$
- 101:  $2^{13}/f_{PSC}$
- 110:  $2^{14}/f_{PSC}$
- 111:  $2^{15}/f_{PSC}$

• **TB1C Register**

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

Bit 7      **TB1ON**: Time Base 1 Control

0: Disable

1: Enable

Bit 6~3    Unimplemented, read as “0”

Bit 2~0    **TB12~TB10**: Select Time Base 1 Time-out Period

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

001:  $2^9/f_{PSC}$

010:  $2^{10}/f_{PSC}$

011:  $2^{11}/f_{PSC}$

100:  $2^{12}/f_{PSC}$

101:  $2^{13}/f_{PSC}$

110:  $2^{14}/f_{PSC}$

111:  $2^{15}/f_{PSC}$

### 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, 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, MF<sub>n</sub>F, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

It is recommended that programs do not use the “CALL” instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a CALL subroutine is executed in the interrupt subroutine.

Every interrupt has the capability of waking up the microcontroller when it is in SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine. To return from an interrupt subroutine, either an RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

## Low Voltage Detector – LVD

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

### LVD Register

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

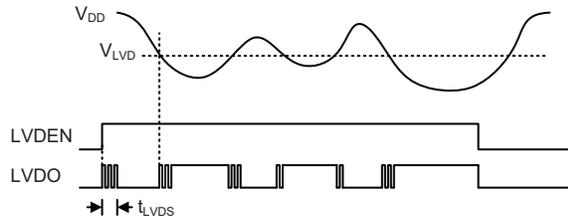
#### • LVDC Register

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

- Bit 7~6      Unimplemented, read as “0”
- Bit 5        **LVDO**: LVD Output Flag  
               0: No Low Voltage Detected  
               1: Low Voltage Detected
- Bit 4        **LVDEN**: Low Voltage Detector Control  
               0: Disable  
               1: Enable
- Bit 3        **VBGEN**: Bandgap Voltage Output Enable Control  
               0: Disable  
               1: Enable
- Bit 2~0     **VLVD2~VLVD0**: Select LVD Voltage  
               000: 2.0V  
               001: 2.2V  
               010: 2.4V  
               011: 2.7V  
               100: 3.0V  
               101: 3.3V  
               110: 3.6V  
               111: 4.0V

### LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage,  $V_{DD}$ , with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.0V and 4.0V. When the power supply voltage,  $V_{DD}$ , falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. When the device is in the IDLE/SLEEP mode the low voltage detector will remain active if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay  $t_{LVDS}$  should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the  $V_{DD}$  voltage may rise and fall rather slowly, at the voltage nears that of  $V_{LVD}$ , there may be multiple bit LVDO transitions.



**LVD Operation**

The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of  $t_{LVD}$  after the LVDO bit has been set high by a low voltage condition. When the device is powered down the Low Voltage Detector will remain active if the LVDEN bit is high. 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 SLEEP or 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 SLEEP or IDLE Mode.

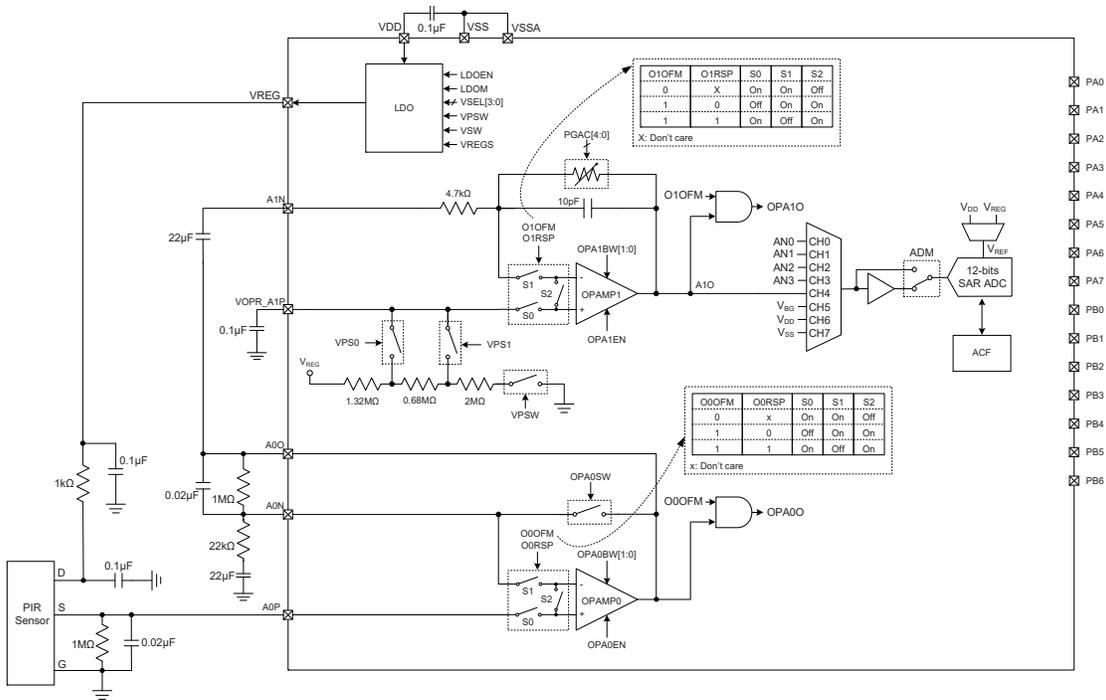
When LVD function is enabled, it is recommended to clear LVF flag first, and then enables interrupt function to avoid mistake action.

## 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. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later using the application program. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
<b>Oscillator Options</b>	
1	HIRC Frequency Selection: 2/4/8MHz
<b>Watchdog Timer Options</b>	
2	WDT function: Always Enable or By WDTC Control

## 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 three 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 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 set 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 following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

### 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
ADDM A,[m]	Add ACC to Data Memory	1 <sup>Note</sup>	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1 <sup>Note</sup>	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1 <sup>Note</sup>	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1 <sup>Note</sup>	Z, C, AC, OV
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].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 or current 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
<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
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-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 up to 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.

3. For the "CLR WDT1" and "CLR WDT2" instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both "CLR WDT1" and "CLR WDT2" instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

## 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
<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
<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
<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
<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
<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>CLR WDT1</b>	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO $\leftarrow$ 0 PDF $\leftarrow$ 0
Affected flag(s)	TO, PDF
<b>CLR WDT2</b>	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.
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 "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 "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 "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 ← Stack ACC ← x
Affected flag(s)	None
<b>RETI</b>	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None
<b>RL [m]</b>	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← [m].7
Affected flag(s)	None
<b>RLA [m]</b>	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← [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) ← [m].i; (i=0~6) [m].0 ← C C ← [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) ← [m].i; (i=0~6) ACC.0 ← C C ← [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 ← [m].(i+1); (i=0~6) [m].7 ← [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	$ACC \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C
<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	$[m] \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C
<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>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

<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
<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
<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	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 or current page) to TBLH and Data Memory
Description	The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no 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>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

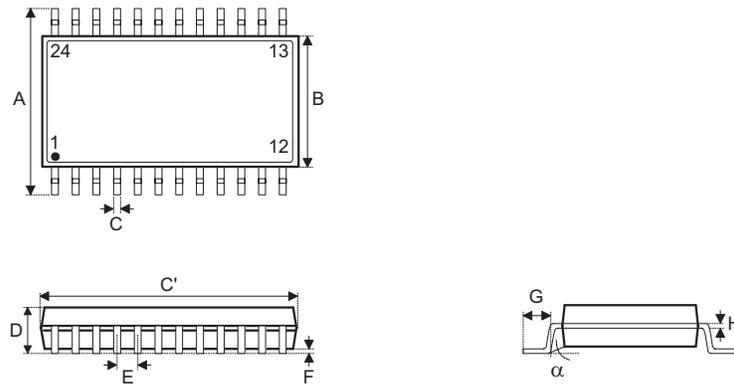
## Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the [Holtek website](#) for the latest version of the [Package/Carton Information](#).

Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

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

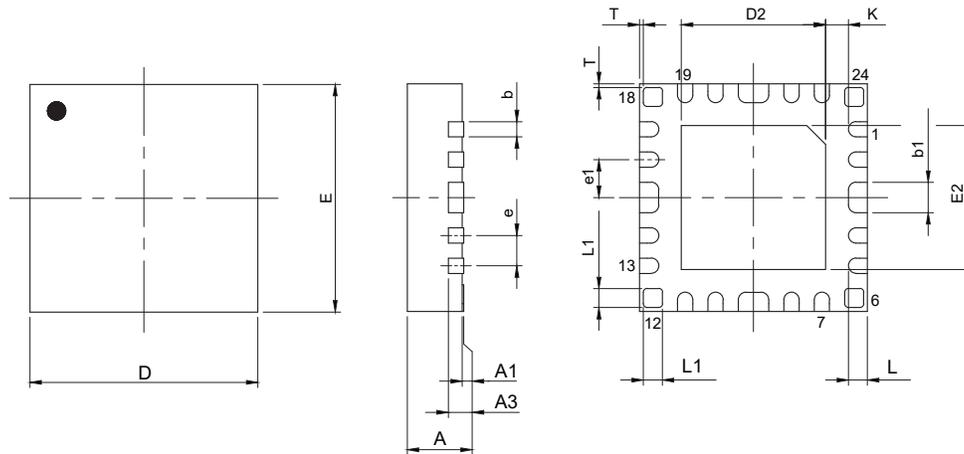
**24-pin SSOP (150mil) Outline Dimensions**



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

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

**SAW Type 24-pin QFN (3mm×3mm×0.55mm) Outline Dimensions**



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.020	0.022	0.024
A1	0.000	0.001	0.002
A3	—	0.006 BSC	—
b	0.006	0.008	0.010
b1	0.014	0.016	0.018
D	—	0.118 BSC	—
E	—	0.118 BSC	—
e	—	0.016 BSC	—
e1	—	0.020 BSC	—
D2	0.073	0.075	0.077
E2	0.073	0.075	0.077
L	0.006	0.010	0.014
L1	0.008	0.010	0.012
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.50	0.55	0.60
A1	0.00	0.02	0.05
A3	—	0.15 BSC	—
b	0.15	0.20	0.25
b1	0.35	0.40	0.45
D	—	3.00 BSC	—
E	—	3.00 BSC	—
e	—	0.40 BSC	—
e1	—	0.50 BSC	—
D2	1.85	1.90	1.95
E2	1.85	1.90	1.95
L	0.15	0.25	0.35
L1	0.20	0.25	0.30
K	0.20	—	—

Copyright© 2019 by HOLTEK SEMICONDUCTOR INC.

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