Body Fat Measurment Flash MCU

BH66F2650/BH66F2660

Revision: V1.10 Date: January 04, 2018

www.holtek.com

Table of Contents

Features

CPU Features

- Operating Voltage
  - f_{SYS}=4MHz: 2.2V~5.5V
  - f_{SYS}=8MHz: 2.2V~5.5V
  - f_{SYS}=12MHz: 2.7V~5.5V
  - f_{SYS}=16MHz: 3.3V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Four Oscillators:
  - High Speed Internal RC – HIRC
  - External 32.768kHz Crystal – LXT
  - High Speed External Crystal – HXT
  - Internal 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal 4MHz, 8MHz or 12MHz oscillator requires no external components
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 8K×16 ~ 16K×16
- RAM Data Memory: 256×8 ~ 1024×8
- True EEPROM Memory: 64×8 ~ 256×8
- Watchdog Timer function
- In Application Programming – IAP
- Up to 28 bidirectional I/O lines
- 2 differential or 4 single-end channels 24-bit resolution Delta Sigma A/D converter
- Two pin-shared external interrupts
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output or single pulse output function
- Serial Interface Module with Dual SPI and I^2C interfaces – SIM
- Single Serial Peripheral Interface – SPIA
- UART Interface for full duplex asynchronous communication
- Dual Time-Base functions for generation of fixed time interrupt signals
- Low Voltage Reset function – LVR
- Low Voltage Detect function – LVD
- Body Fat Circuit
- 16-bit Multiplication Division Unit
- Package type: 48-pin LQFP

General Description

This series of devices are specifically designed for eight-electrode AC Body Fat Scale applications. Measuring body fat uses a technique whereby an AC current flowing through the human body is measured and then used to calculate a body fat value. The specialised circuits to do this are a weight measurement circuit and a fat measurement circuit. The weight measurement circuit uses an external load cell to output a signal, which after amplification by an operational amplifier, and then conversion using an A/D converter, reads the corresponding value as the calculated weight. The fat measurement circuit uses an AC signal via an electrode slice to flow through human body. After amplification by an internal operational amplifier, and then conversion by an A/D converter, the measured value is one representing body impedance, which is used to calculate the corresponding body fat value.

This series of devices are Flash Memory I/O type 8-bit high performance RISC architecture microcontroller which includes a multi-channel 24-bit Delta Sigma A/D converter, designed for applications that interface directly to analog signals and which require a low noise and high accuracy analog-to-digital converter. Offering users the convenience of Flash Memory multi-programming features, the devices also includes a 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 24-bit Delta Sigma A/D converter, PGA and LDO and other circuitry specifically designed for Body Fat Scale applications. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is provided by including fully integrated SPI, I^2C and UART interface functions, popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of external, internal and high and low oscillators functions are 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 devices also include a multiplication/division unit. 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 and reductions in circuit board areas.

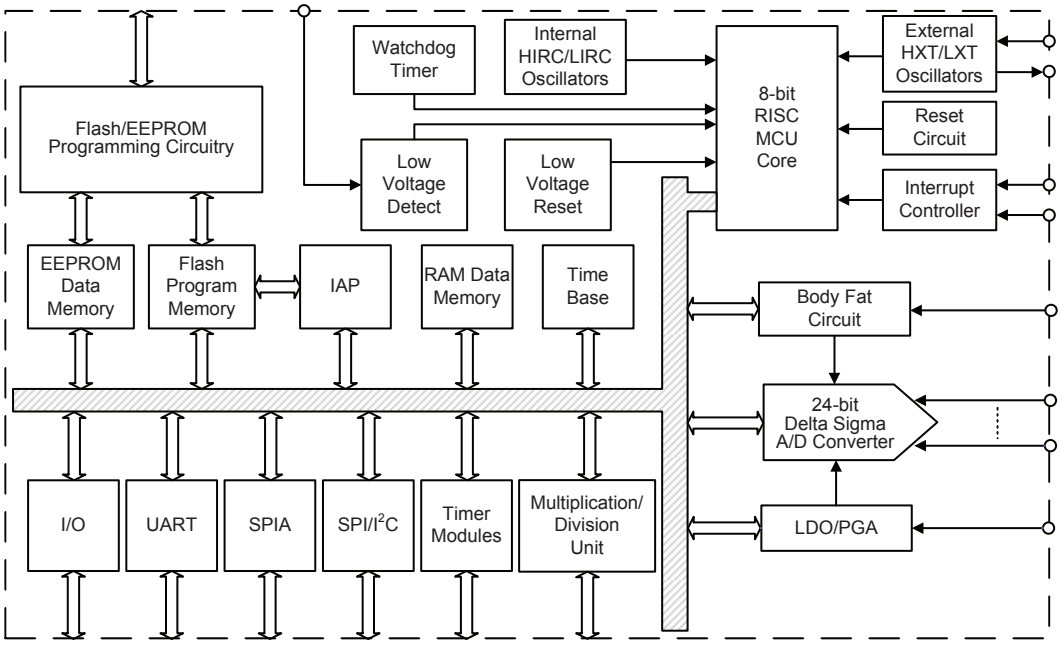
Selection Table

Most features are common to both devices. The main features distinguishing them are Memory capacity. The following table summarises the main features of each device.

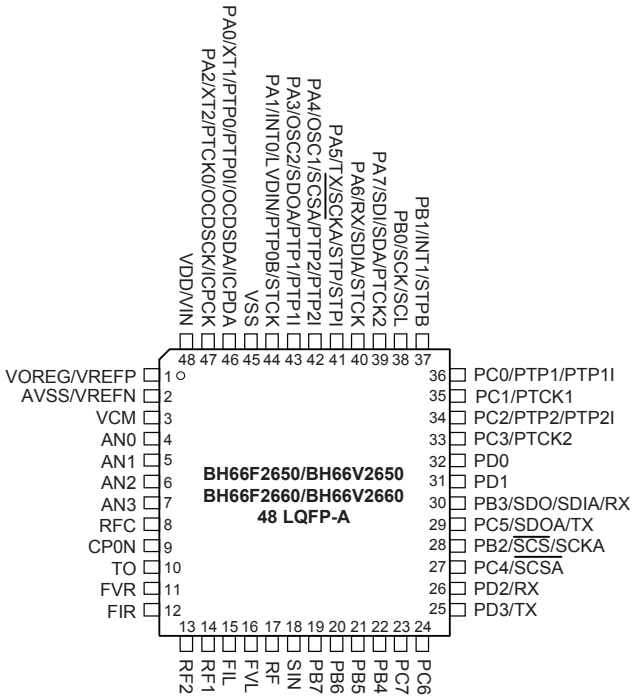
Part No.	V_{DD}	ROM	RAM	EEPROM	I/O	Ext.Int.	A/D Converter
BH66F2650	2.2V~5.5V	8K×16	256×8	64×8	28	2	24-bit×4
BH66F2660	2.2V~5.5V	16K×16	1024×8	256×8	28	2	24-bit×4

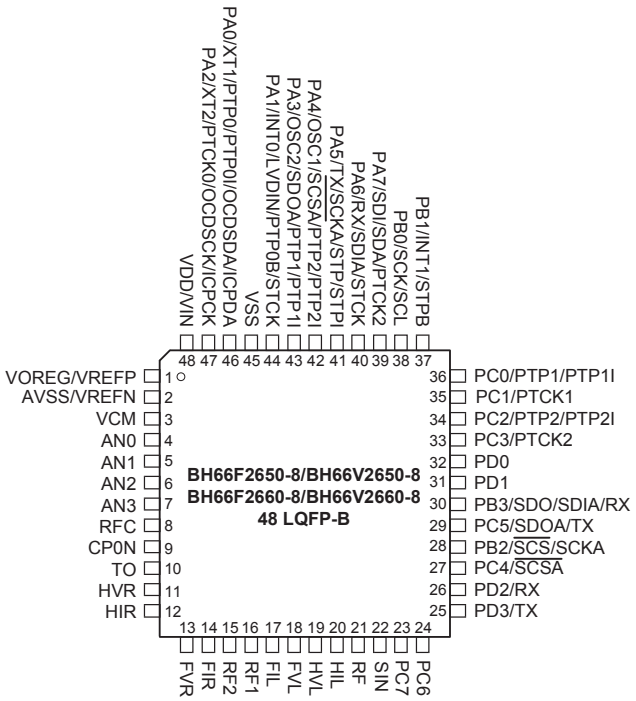
Part No.	Timer Module	Interface	MDU	Time Base	Stack	Package
BH66F2650	10-bit PTM×3 16-bit STM×1	SIM/UART/SPIA	√	2	8	48LQFP
BH66F2660	10-bit PTM×3 16-bit STM×1	SIM/UART/SPIA	√	2	8	48LQFP

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 OCDSDA and OCDSCK pins are supplied as dedicated OCDS pins and as such only available for the BH66V2650/BH66V2660 devices which are the OCDS EV chip for the BH66F2650/BH66F2660 devices.

Pin Description

Pin Name	Function	OPT	I/T	O/T	Descriptions
PA0/XT1/PTP0/PTP0I/ OCDSDA/ICPDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	XT1	PAS0	LXT	—	LXT input pin
	PTP0	PAS0	—	CMOS	PTM0 output
	PTP0I	PAS0	ST	—	PTM0 capture input
	OCDSDA	—	ST	CMOS	OCDS address/data line, for EV chip only
	ICPDA	—	ST	CMOS	ICP address/data line
PA1/INT0/LVDIN/ PTP0B/STCK	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT0	PAS0 INTEG INTC0	ST	—	External interrupt 0
	LVDIN	PAS0	AN	—	LVD input
	PTP0B	PAS0	—	CMOS	PTM0 inverting output
	STCK	PAS0 IFS0	ST	—	STM clock input
PA2/XT2/PTCK0/ OCDSCK/ICPCK	PA2	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	XT2	PAS0	—	LXT	LXT output pin
	PTCK0	PAS0	ST	—	PTM0 clock input
	OCDSCK	—	ST	—	OCDS clock input, for EV chip only
	ICPCK	—	ST	—	ICP clock line
PA3/OSC2/SDOA/ PTP1/PTP1I	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OSC2	PAS0	—	HXT	HXT output pin
	SDOA	PAS0	—	CMOS	SPIA serial data output
	PTP1	PAS0	—	CMOS	PTM1 output
	PTP1I	PAS0 IFS0	ST	—	PTM1 capture input
PA4/OSC1/$\overline{SCSA}$/ PTP2/PTP2I	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OSC1	PAS1	HXT	—	HXT input pin
	$\overline{SCSA}$	PAS1 IFS1	ST	CMOS	SPIA slave select pin
	PTP2	PAS1	—	CMOS	PTM2 output
	PTP2I	PAS1 IFS0	ST	—	PTM2 capture input
PA5/TX/SCKA/STP/ STPI	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TX	PAS1	—	CMOS	UART TX serial data output pin
	SCKA	PAS1 IFS1	ST	—	SPIA serial clock input
	STP	PAS1	—	CMOS	STM output
	STPI	PAS1	ST	—	STM capture input

Pin Name	Function	OPT	I/T	O/T	Descriptions
PA6/RX/SDIA/STCK	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	RX	PAS1 IFS1	ST	—	UART RX serial data input pin
	SDIA	PAS1 IFS1	ST	—	SPIA serial data input
	STCK	PAS1 IFS0	ST	—	STM clock input
PA7/SDI/SDA/PTCK2	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDI	PAS1	ST	—	SPI (SIM) serial data input
	SDA	PAS1	ST	NMOS	I2C (SIM) data line
	PTCK2	PAS1 IFS0	ST	—	PTM2 clock input
PB0/SCK/SCL	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	SCK	PBS0	ST	CMOS	SPI (SIM) serial clock
	SCL	PBS0	ST	NMOS	I2C (SIM) clock line
PB1/INT1/STPB	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	INT1	PBS0 INTEG INTC0	ST	—	External interrupt 1
	STPB	PBS0	—	CMOS	STM inverting output
PB2/$\overline{SCS}$/SCKA	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	$\overline{SCS}$	PBS0	ST	CMOS	SPI (SIM) slave chip select
	SCKA	PBS0 IFS1	ST	CMOS	SPIA serial clock input
PB3/SDO/SDIA/RX	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	SDO	PBS0	—	CMOS	SPI (SIM) serial data output
	SDIA	PBS0 IFS1	ST	—	SPIA serial data input
	RX	PBS0 IFS1	ST	—	UART RX serial data input pin
PB4~PB7	PBn	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
PC0/PTP1/PTP1I	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	PTP1	PCS0	—	CMOS	PTM1 output
	PTP1I	PCS0 IFS0	ST	—	PTM1 capture input
PC1/PTCK1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	PTCK1	—	ST	—	PTM1 clock input
PC2/PTP2/PTP2I	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	PTP2	PCS0	—	CMOS	PTM2 output
	PTP2I	PCS0 IFS0	ST	—	PTM2 capture input

Pin Name	Function	OPT	I/T	O/T	Descriptions
PC3/PTCK2	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	PTCK2	—	ST	—	PTM2 clock input
PC4/$\overline{SCSA}$	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high.
	$\overline{SCSA}$	PCS1 IFS1	ST	CMOS	SPIA slave select pin
PC5/SDOA/TX	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high.
	SDOA	PAS0 PCS1	—	CMOS	SPIA serial data output
	TX	PCS1	—	CMOS	UART TX serial data output pin
PC6~ PC7	PCn	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
PD0~ PD1	PDn	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
PD2/RX	PD2	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	RX	PDS0 IFS1	ST	—	UART RX serial data input pin
PD3/TX	PD3	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	TX	PDS0	—	CMOS	UART TX serial data output pin
A/D Converter					
VOREG/VREFP	VOREG	—	PWR	—	LDO output pin
			PWR	—	Positive power supply for VCM, A/D converter, PGA
	VREFP	—	AN	—	External positive reference input of A/D converter
AVSS/VREFN	AVSS	—	PWR	—	Negative power supply for VCM, A/D converter, PGA
	VREFN	—	AN	—	External negative reference input of A/D converter
AN0 ~ AN3	ANn	—	AN	—	A/D input pin
VCM	VCM	—	AN	—	External input voltage for A/D converter common mode
			—	AN	A/D converter Common mode voltage output
Body Fat Circuit					
RFC	RFC	—	AN	—	A/D converter analog input
CP0N	CP0N	—	AN	—	Peak detector input
TO	TO	—	—	AN	Operational amplifier output
HVR	HVR	—	AN	AN	Right hand channel 1
HIR	HIR	—	AN	AN	Right hand channel 2
FVR	FVR	—	AN	AN	Right foot channel 1
FIR	FIR	—	AN	AN	Right foot channel 2
RF2	RF2	—	AN	AN	Reference 2 impedance channel
RF1	RF1	—	AN	AN	Reference 1 impedance channel
FIL	FIL	—	AN	AN	Left foot channel 2
FVL	FVL	—	AN	AN	Left foot channel 1
HVL	HVL	—	AN	AN	Left hand channel 1
HIL	HIL	—	AN	AN	Left hand channel 2
RF	RF	—	AN	AN	Reference 1/2 impedance channel
SIN	SIN	—	—	AN	Sine wave output

Pin Name	Function	OPT	I/T	O/T	Descriptions
Power					
VDD/VIN	VDD	—	PWR	—	Positive power supply
	VIN	—	PWR	—	LDO input pin
VSS	VSS	—	PWR	—	Negative power supply.

Legend: I/T: Input type;
OPT: Optional by register option;
ST: Schmitt Trigger input;
NMOS: NMOS output;
HXT: High frequency crystal oscillator;
LXT: Low frequency crystal oscillator
O/T: Output type
PWR: Power
CMOS: CMOS output
AN: Analog signal

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}-0.3V~V_{SS}+6.0V
Input Voltage ..... V_{SS}-0.3V to V_{DD}+0.3V
Storage Temperature ..... -50°C to 125°C
Operating Temperature ..... -40°C to 85°C
I_{OH} Total ..... -80mA
I_{OL} Total 80mA
Total Power Dissipation 500mW

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

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=-40°C ~ 85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HXT	f_{SYS}=4MHz	2.2	—	5.5	V
		f_{SYS}=8MHz	2.2	—	5.5	
		f_{SYS}=12MHz	2.7	—	5.5	
		f_{SYS}=16MHz	3.3	—	5.5	
	Operating Voltage – HIRC	f_{SYS}=4MHz	2.2	—	5.5	V
		f_{SYS}=8MHz	2.2	—	5.5	
		f_{SYS}=12MHz	2.7	—	5.5	
	Operating Voltage – LXT	f_{SYS}=32.768kHz	2.2	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32kHz	2.2	—	5.5	V

Standby Current Characteristics

Ta=25°C

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max.	Unit
		V_{DD}	Conditions				85°C	
I_{STB}	SLEEP Mode	2.2V	WDT off	—	0.2	0.6	0.7	μA
		3V		—	0.2	0.8	1.0	
		5V		—	0.5	1.0	1.2	
		2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3.0	5.0	6.0	
	IDLE0 Mode – LIRC	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3.0	5.0	6.0	
		5V		—	5.0	10	12	
	IDLE0 Mode – LXT	2.2V	f_{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} on, f_{SYS}=4MHz	—	0.17	0.24	0.26	mA
		3V		—	0.25	0.35	0.385	
		5V		—	0.48	0.67	0.70	
		2.2V	f_{SUB} on, f_{SYS} =8MHz	—	0.32	0.45	0.48	mA
		3V		—	0.47	0.66	0.70	
		5V		—	0.91	1.27	1.30	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	0.60	0.84	0.87	mA
		3V		—	0.68	0.95	1.00	
		5V		—	1.33	1.86	1.90	
	IDLE1 Mode – HXT	2.2V	f_{SUB} on, f_{SYS} =4MHz	—	0.17	0.24	0.26	mA
		3V		—	0.25	0.35	0.385	
		5V		—	0.48	0.67	0.70	
		2.2V	f_{SUB} on, f_{SYS}=8MHz	—	0.32	0.45	0.48	mA
		3V		—	0.47	0.66	0.70	
		5V		—	0.91	1.27	1.30	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	0.60	0.84	0.87	mA
		3V		—	0.68	0.95	1.00	
		5V		—	1.33	1.86	1.90	
		3.3V	f_{SUB} on, f_{SYS}=16MHz	—	1.10	1.60	1.90	mA
		5V		—	1.80	2.50	2.60	

Notes: 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=25°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2.2V	f_{SYS} = 32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	SLOW Mode – LXT	2.2V	f_{SYS} = 32768Hz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	2.2V	f_{SYS} = 4MHz	—	0.3	0.5	mA
		3V		—	0.4	0.6	
		5V		—	0.8	1.2	
		2.2V	f_{SYS} = 8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	
		2.7V	f_{SYS} = 12MHz	—	1.0	1.4	mA
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	
	FAST Mode – HXT	2.2V	f_{SYS} = 4MHz	—	0.4	0.6	mA
		3V		—	0.5	0.75	
		5V		—	1.0	1.5	
		2.2V	f_{SYS} = 8MHz	—	0.8	1.2	mA
		3V		—	1.0	1.5	
		5V		—	2.0	3.0	
		2.7V	f_{SYS} = 12MHz	—	1.2	2.2	mA
		3V		—	1.5	2.75	
		5V		—	3.0	4.5	
		3.3V	f_{SYS} = 16MHz	—	3.2	4.8	mA
		5V		—	4.0	6.0	

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

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

A.C. Characteristics

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

High Speed Internal Oscillator – HIRC – Frequency Accuracy

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

Symbol	Parameter	Test Conditions		Min	Typ	Max	Unit
		V_{DD}	Temp.				
f_{HIRC}	4MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	4	+1%	MHz
			-40°C ~ 85°C	-2%	4	+2%	
		2.2V~5.5V	25°C	-2.5%	4	+2.5%	
			-40°C ~ 85°C	-3%	4	+3%	
	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C ~ 85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C ~ 85°C	-3%	8	-3%	
	12MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	12	+1%	MHz
			-40°C ~ 85°C	-2%	12	+2%	
		2.7V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C ~ 85°C	-3%	12	+3%	

Notes: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

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

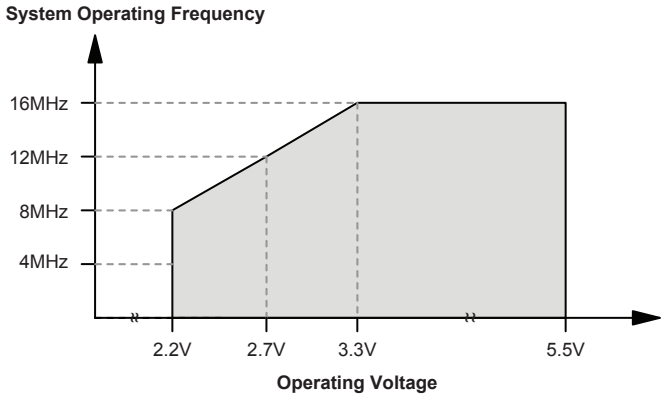
3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will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

Low Speed Internal Oscillator Characteristics – LIRC

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	2.2V~5.5V	25°C	-5%	32	+5%	kHz
			-40°C~85°C	-10%	32	+10%	

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time Wake-up from condition where f_{SYS} is off	—	$f_{SYS} = f_H \sim f_H/64$, $f_H = f_{HXT}$	—	128	—	t_{HXT}
		—	$f_{SYS} = f_H \sim f_H/64$, $f_H = f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS} = f_{SUB} = f_{LXT}$	—	1024	—	t_{LXT}
		—	$f_{SYS} = f_{SUB} = f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time Wake-up from condition where f_{SYS} is on	—	$f_{SYS} = f_H \sim f_H/64$, $f_H = f_{HXT}$ or f_{HIRC}	—	2	—	t_H
		—	$f_{SYS} = f_{SUB} = f_{LXT}$ or f_{LIRC}	—	2	—	t_{SUB}
	System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode	—	f_{HXT} switches from off → on	—	1024	—	t_{HXT}
		—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
		—	f_{LXT} switches from off → on	—	1024	—	t_{LXT}
t_{RSTD}	System Reset Delay Time Reset source from Power-on reset or LVR hardware reset	—	RR_{POR}=5V/ms	42	48	54	ms
	System Reset Delay Time LVRC/WDTC/RSTC software reset	—	—				
	System Reset Delay Time Reset source from WDT overflow	—	—	14	16	18	ms
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

Notes: 1.For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.

2. The time units, shown by the symbols t_{HXT}, t_{HIRC} etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example $t_{HIRC} = 1/f_{HIRC}$, $t_{SYS} = 1/f_{SYS}$ etc.
3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.
4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports	5V	—	3.5	—	5.0	V
		—		$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Ports	3V	V_{OL}=0.1V_{DD}	16	32	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Ports	3V	V_{OH} = 0.9V_{DD}, SLEDCn[m+1, m] = 00B (n = 0,1; m = 0 or 2 or 4 or 6)	-0.7	-1.5	—	mA
		5V	V_{OH} = 0.9V_{DD}, SLEDCn[m+1, m] = 00B (n = 0,1; m = 0 or 2 or 4 or 6)	-1.5	-2.9	—	
		3V	V_{OH} = 0.9V_{DD}, SLEDCn[m+1, m] = 01B (n = 0,1;m = 0 or 2 or 4 or 6)	-1.3	-2.5	—	
		5V	V_{OH} = 0.9V_{DD}, SLEDCn[m+1, m] = 01B (n = 0,1; m = 0 or 2 or 4 or 6)	-2.5	-5.1	—	
		3V	V_{OH} = 0.9V_{DD}, SLEDCn[m+1, m] = 10B (n = 0,1; m = 0 or 2 or 4 or 6)	-1.8	-3.6	—	
		5V	V_{OH} = 0.9V_{DD}, SLEDCn[m+1, m] = 10B (n = 0,1;m = 0 or 2 or 4 or 6)	-3.6	-7.3	—	
		3V	V_{OH} = 0.9V_{DD}, SLEDCn[m+1, m] = 11B (n = 0,1; m = 0 or 2 or 4 or 6)	-4	-8	—	
		5V	V_{OH} = 0.9V_{DD}, SLEDCn[m+1, m] = 11B (n = 0,1;m = 0 or 2 or 4 or 6)	-8	-16	—	
V_{OL}	Output Low Voltage for I/O Ports	3V	I_{OL} = 16mA	—	—	0.3	V
		5V	I_{OL} = 32mA	—	—	0.5	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{OH}	Output High Voltage for I/O Ports	3V	I_{OH} = -0.7mA, SLEDCn[m+1, m] = 00B (n = 0,1; m = 0 or 2 or 4 or 6)	2.7	—	—	V
		5V	I_{OH} = -1.5mA, SLEDCn[m+1, m] = 00B (n = 0,1;m = 0 or 2 or 4 or 6)	4.5	—	—	
		3V	I_{OH} = -1.3mA, SLEDCn[m+1, m] = 01B (n = 0,1; m = 0 or 2 or 4 or 6)	2.7	—	—	
		5V	I_{OH} = -2.5mA, SLEDCn[m+1, m] = 01B (n = 0,1; m = 0 or 2 or 4 or 6)	4.5	—	—	
		3V	I_{OH} = -1.8mA, SLEDCn[m+1, m] = 10B (n = 0,1; m = 0 or 2 or 4 or 6)	2.7	—	—	
		5V	I_{OH} = -3.6mA, SLEDCn[m+1, m] = 10B (n = 0,1; m = 0 or 2 or 4 or 6)	4.5	—	—	
		3V	I_{OH} = -4mA, SLEDCn[m+1, m] = 11B (n = 0,1; m = 0 or 2 or 4 or 6)	2.7	—	—	
		5V	I_{OH} = -8mA, SLEDCn[m+1, m] = 11B (n = 0,1; m = 0 or 2 or 4 or 6)	4.5	—	—	
R_{PH}	Pull-High Resistance for I/O Ports (Note)	3V	—	20	60	100	kΩ
		5V		10	30	50	
I_{LEAK}	Input Leakage Current	3V/5V	V_{IN} = V_{DD} or V_{IN} = V_{SS}	—	—	±1	μA
t_{TPI}	STPI, PTPnI Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TCK}	STCK, PTCKn Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs

Note: The R_{PH} internal pull high resistance value is calculated by connecting to ground and enabling the input pin with a pull-high resistor and then measuring the input sink current at the specified supply voltage level. Dividing the voltage by this measured current provides the R_{PH} value.

Memory Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{RW}	V_{DD} for Read / Write	—	—	V_{DDmin}	—	V_{DDmax}	V
Program Flash / Data EEPROM Memory							
t_{DEW}	Erase / Write Cycle Time	—	—	2.2	2.5	2.8	ms
I_{DDPGM}	Programming / Erase Current on V_{DD}	—	—	—	—	5.0	mA
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta = 25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

LVD/LVR Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.10V	-5%	2.1	+5%	V
		—	LVR enable, voltage select 2.55V	-5%	2.55	+5%	
		—	LVR enable, voltage select 3.15V	-5%	3.15	+5%	
		—	LVR enable, voltage select 3.80V	-5%	3.8	+5%	
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 1.04V	-10%	1.04	+10%	V
		—	LVD enable, voltage select 2.20V	-5%	2.2	+5%	
		—	LVD enable, voltage select 2.40V	-5%	2.4	+5%	
		—	LVD enable, voltage select 2.70V	-5%	2.7	+5%	
		—	LVD enable, voltage select 3.00V	-5%	3.0	+5%	
		—	LVD enable, voltage select 3.30V	-5%	3.3	+5%	
		—	LVD enable, voltage select 3.60V	-5%	3.6	+5%	
		—	LVD enable, voltage select 4.00V	-5%	4.0	+5%	
$I_{LVRLVDBG}$	Operating Current	3V	LVD enable, LVR enable, VBGEN=0	—	—	18	μA
		5V	LVD enable, LVR enable, VBGEN=0	—	20	25	
		3V	LVD enable, LVR enable, VBGEN=1	—	—	150	
		5V	LVD enable, LVR enable, VBGEN=1	—	25	30	
t_{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	15	μs
		—	For LVR disable, VBGEN=0, LVD off → on	—	—	150	
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs
I_{LVR}	Additional Current for LVR Enable	—	LVD disable, VBGEN=0	—	—	24	μA
I_{LVD}	Additional Current for LVD Enable	—	LVR disable, VBGEN=0	—	—	24	μA

24-bit A/D Converter Electrical Characteristics

LDO + VCM Test conditions: MCU HALT, other function disable;
$V_{DD}=V_{IN}$, Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IN}	LDO Input Voltage	—	—	2.6	—	5.5	V
I_Q	LDO Quiescent Current	—	LDOVS[1:0]=00B, V_{IN} =3.6V, No load	—	400	520	μA
V_{OUT_LDO}	LDO Output Voltage	—	LDOVS[1:0]=00B, V_{IN}=3.6V, I_{LOAD}=0.1mA	-5%	2.4	+ 5%	V
		—	LDOVS[1:0]=01B, V_{IN}=3.6V, I_{LOAD}=0.1mA		2.6		
		—	LDOVS[1:0]=10B, V_{IN}=3.6V, I_{LOAD}=0.1mA		2.9		
		—	LDOVS[1:0]=11B, V_{IN}=3.6V, I_{LOAD}=0.1mA		3.3		
ΔV_{LOAD}	LDO Load Regulation(Note 1)	—	LDOVS[1:0]=00B, $V_{IN}=V_{OUT\_LDO}$+0.2V, 0mA ≤ I_{LOAD} ≤ 10mA	—	0.105	0.21	%/mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DROP_LDO}	LDO Dropout Voltage(Note2)	—	LDOVS[1:0]=00B, V_{IN}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}=2%	—	—	220	mV
		—	LDOVS[1:0]=01B, V_{IN}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}=2%	—	—	200	
		—	LDOVS[1:0]=10B, V_{IN}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}=2%	—	—	180	
		—	LDOVS[1:0]=11B, V_{IN}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}=2%	—	—	160	
TC_{LDO}	LDO Temperature Coefficient	—	Ta=-40°C~85°C, LDOVS[1:0]=00B, V_{IN}=3.6V, I_{LOAD}=100μA	—	—	0.48	mV/°C
ΔV_{LINE_LDO}	LDO Line Regulation	—	LDOVS[1:0]=00B, 2.6V ≤ V_{IN} ≤ 5.5V, I_{LOAD}=100μA	—	—	0.7	%/V
			LDOVS[1:0]=00B, 2.6V ≤ V_{IN} ≤ 3.6V, I_{LOAD}=100μA	—	—	0.2	%/V
V_{OUT_VCM}	VCM Output Voltage	—	V_{OREG}=3.3V, No load	-5%	1.25	+ 5%	V
TC_{VCM}	VCM Temperature Coefficient	—	Ta=-40°C~85°C, V_{OREG}=3.3V, I_{LOAD}=10μA	—	—	0.24	mV/°C
ΔV_{LINE_VCM}	VCM Line Regulation	—	2.4V ≤ V_{OREG} ≤ 3.3V, No load	—	—	0.3	%/V
t_{VCMS}	VCM Turn on Stable Time	—	V_{OREG}=3.3V, No load	—	—	10	ms
I_{OH}	Source Current for VCM Output Pin	—	V_{OREG}=3.3V, ΔV_{OUT_VCM}=-2%	1	—	—	mA
I_{OL}	Sink Current for VCM Output Pin	—	V_{OREG}=3.3V, ΔV_{OUT_VCM}=+2%	1	—	—	mA
A/D Converter & A/D Converter Internal Reference Voltage (Delta Sigma A/D Converter)							
V_{OREG}	Supply Voltage for VCM, A/D converter, PGA, OPA	—	LDOEN=0	2.4	—	3.3	V
		—	LDOEN=1	2.4	—	3.3	
I_{ADC}	Additional Current for A/D Converter Enable	—	VCM enable, VRBUFP=1 and VRBUFN=1	—	750	900	μA
		—	VCM enable, VRBUFP=0 and VRBUFN=0	—	600	750	
I_{ADSTB}	Standby Current	—	MCU enters SLEEP mode, No Load	—	—	1	μA
N_R	Resolution	—	—	—	—	24	Bit
INL	Integral Non-linearity	—	V_{OREG}=3.3V, V_{REF}=1.25V, ΔSI=±450mV, PGA gain=1	—	±50	±200	ppm
NFB	Noise Free Bits	—	PGA gain=128, Data rate=10Hz	—	15.4	—	Bit
ENOB	Effective Number of Bits	—	PGA gain=128, Data rate=10Hz	—	18.1	—	Bit
f_{ADCK}	A/D Converter Clock Frequency	—	—	40	409.6	440	kHz
f_{ADO}	A/D Converter Output Data Rate	—	f_{MCLK}=4MHz, FLMS[2:0]=000B	4	—	521	Hz
		—	f_{MCLK}=4MHz, FLMS[2:0]=010B	10	—	1302	
V_{REFP}	External Reference Input Voltage	—	VREFS=1, VRBUFP=0, VRBUFN=0	V_{REFN} + 0.8	—	V_{OREG}	V
V_{REFN}		—		0	—	V_{REFP} - 0.8	V
V_{REF}		—	VGS[1:0]=00B, V_{REF}=V_{REFP}−V_{REFN}	0.80	—	1.75	V
PGA							
V_{CM_PGA}	Common Mode Voltage Range	—	—	0.4	—	V_{OREG} - 0.95	V
ΔD_I	Differential Input Voltage Range	—	Gain=PGS × AGS	-V_{REF} /Gain	—	+V_{REF} /Gain	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Temperature Sensor							
TC_{TS}	Temperature Sensor Temperature Coefficient	—	Ta=-40°C~85°C, V_{REF}=1.25V, VGS[1:0]=00B (Gain=1), VRBUFP=0, VRBUFN=0	—	175	—	μV/°C

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

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

12-bit D/A Converter Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DACO}	Output Voltage Range	—	—	V_{SS}	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	1.05	—	V_{DD}	V
I_{DAC}	Additional Current for D/A Converter Enable	—	V_{REF}=5V	—	—	450	μA
DNL	Differential Non-linearity	—	2.4V ≤ V_{DD} ≤ 5.5V	—	—	±4	LSB
INL	Integral Non-linearity	—	2.4V ≤ V_{DD} ≤ 5.5V	—	—	±8	LSB

Operational Amplifier Electrical Characteristics (Body Fat Circuit)

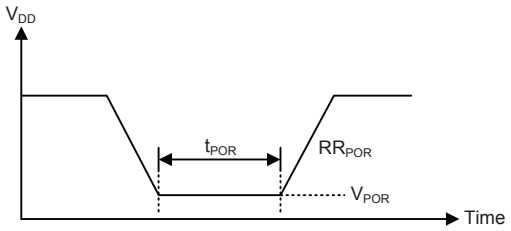
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Supply Voltage	—	—	2.4	—	5.5	V
I_{CC}	Supply Current Per Signal Amplifier	5V	Io=0A	150	360	500	μA
OP0, OP1, OP2							
SR	Slew Rate at Unity Gain	3V	R_L=100kΩ, C_L=100pF	7.5	—	—	V/μs
GBW	Gain Bandwidth Product	3V	R_L=100kΩ, C_L=100pF	—	—	2	MHz

Power-on Reset Characteristics

Ta=25°C

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

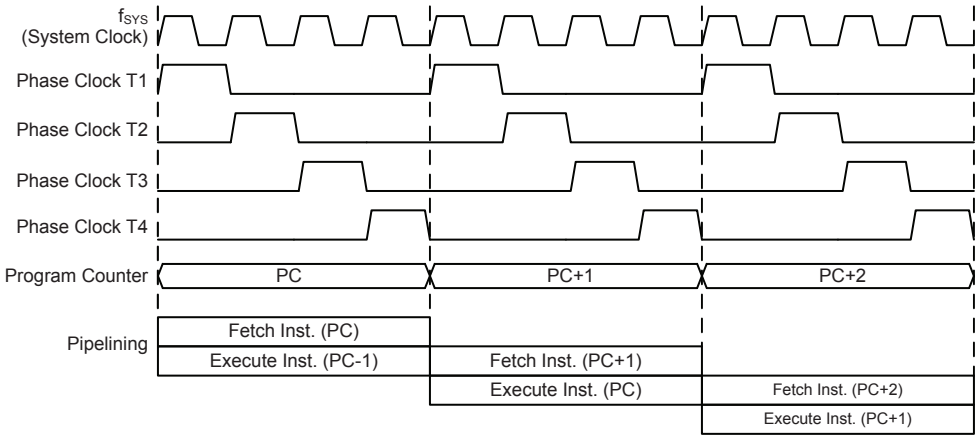


System Architecture

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

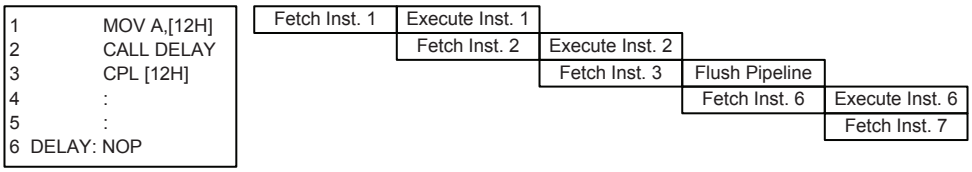
Clocking and Pipelining

The main system clock, derived from either a HXT, LXT, 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.



System Clocking and Pipelining

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.



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.

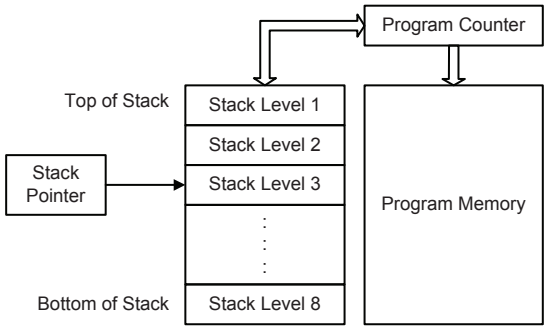
Device	Program Counter	
	Program Counter High Byte	PCL Register
BH66F2650	PC12~PC8	PCL7~PCL0
BH66F2660	PBP0, PC12~PC8	PCL7~PCL0

Program Counter

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

Stack

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



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

Arithmetic and Logic Unit – ALU

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

- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
  LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA

- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
  LRRA, LRR, LRRCA, LRRC, LRLA, LRL, LRLCA, LRLC
- Increment and Decrement:
  INCA, INC, DECA, DEC, LINCA, LINC, LDECA, LDEC
- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI,
  LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

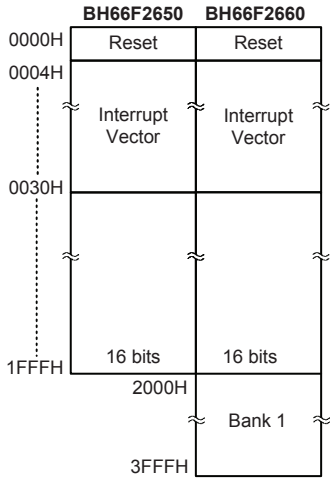
Flash Program Memory

The Program Memory is the location where the user code or program is stored. For the BH66F2650/ BH66F2660 devices the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, the Flash devices offer users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

Structure

The Program Memory has a capacity of 8K×16 ~ 16K×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.

Device	Capacity	Banks
BH66F2650	8K×16	—
BH66F2660	16K×16	0~1



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

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

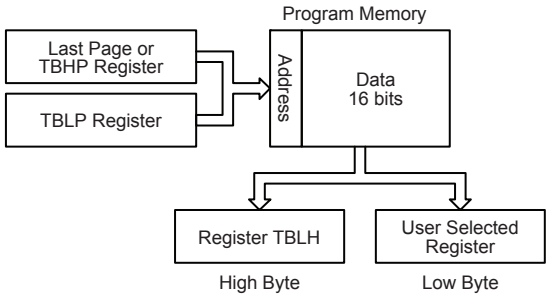


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "1F00H" which refers to the start address of the last page within the 8K Program Memory of the BH66F2650 device. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "1F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the address specified by TBLP and TBHP if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to "0" will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

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

Table Read Program Example

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

In Circuit Programming – ICP

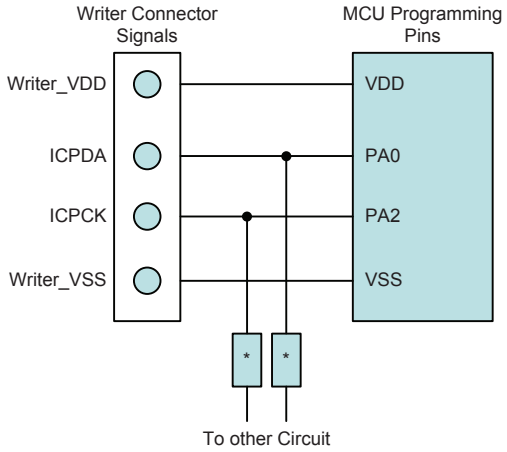
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, a means of programming the microcontroller in-circuit has provided using a 4-pin interface. This provides manufacturers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

The Flash MCU to Writer Programming Pin correspondence table is as follows:

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 and EEPROM data Memory can both 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 ICPCK 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

There is an EV chip named BH66V2650/BH66V2660 which is used to emulate the real MCU device named BH66F2650/BH66F2660. 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 OCDSDA and OCDSCK pins to the HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the 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.

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

In Application Programming – IAP

The devices offer IAP function to update data or application program to Fash ROM. Users can define any ROM location for IAP, but there are some features which user must notice in using IAP function.

IAP Features	BH66F2650 Configurations	BH66F2660 Configurations
Erase Page	32 words / page	64 words / page
Writing Word	32 words / time	64 words / time
Reading Word	1 word / time	1 word / time

In Application Programming Control Registers

The Address registers, FARL and FARH, the Data registers, FD0L/FD0H, FD1L/FD1H, FD2L/FD2H and FD3L/FD3H, and the Control registers, FC0, FC1 and FC2, are the corresponding Flash access registers located in Data Memory sector 1 for IAP. If using the indirect addressing method to access the FC0, FC1 and FC2 registers, all read and write operations to the registers must be performed using the Indirect Addressing Register, IAR1 or IAR2, and the Memory Pointer pair, MP1L/MP1H or MP2L/MP2H. Because the FC0, FC1 and FC2 control registers are located at the address of 43H~45H in Data Memory sector 1, the desired value ranged from 43H to 45H must first be written into the MP1L or MP2L Memory Pointer low byte and the value "01H" must also be written into the MP1H or MP2H Memory Pointer high byte.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	—	CLWB
FARL	A7	A6	A5	A4	A3	A2	A1	A0
FARH (BH66F2650)	—	—	—	A12	A11	A10	A9	A8
FARH (BH66F2660)	—	—	A13	A12	A11	A10	A9	A8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Registers List

- **FC0 Register**

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

Bit 7 **CFWEN**: Flash Memory Write Enable Control
0: Flash memory write function is disabled
1: Flash memory write function has been successfully enabled
When this bit is cleared to "0" by application program, the Flash memory write function is disabled. Note that writing a "1" into this bit results in no action. This bit is used to indicate that the Flash memory write function status. When this bit is set to "1" by hardware, it means that the Flash memory write function is enabled successfully. Otherwise, the Flash memory write function is disabled as the bit content is "0".

Bit 6~4 **FMOD2~FMOD0**: Mode Selection
000: Write program memory
001: Page erase program memory
010: Reserved
011: Read program memory
100: Reserved
101: Reserved
110: FWEN mode – Flash memory Write function Enable mode
111: Reserved

Bit 3 **FWPEN**: Flash Memory Write Procedure Enable Control
0: Disable
1: Enable
When this bit is set to "1" and the FMOD field is set to "110", the IAP controller will execute the "Flash memory write function enable" procedure. Once the Flash memory write function is successfully enabled, it is not necessary to set the FWPEN bit any more.

Bit 2 **FWT**: Flash Memory Write Initiate Control
0: Do not initiate Flash memory write or Flash memory write process is completed
1: Initiate Flash memory write process
This bit is set by software and cleared by hardware when the Flash memory write process is completed.

Bit 1 **FRDEN**: Flash Memory Read Enable Control
0: Flash memory read disable
1: Flash memory read enable

Bit 0 **FRD**: Flash Memory Read Initiate Control
0: Do not initiate Flash memory read or Flash memory read process is completed
1: Initiate Flash memory read process
This bit is set by software and cleared by hardware when the Flash memory read process is completed.

- **FC1 Register**

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

Bit 7~0 **D7~D0**: Whole Chip Reset Pattern
When user writes a specific value of "55H" to this register, it will generate a reset signal to reset whole chip.

- **FC2 Register**

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

Bit 7~1 Unimplemented, read as "0".

Bit 0 **CLWB**: Flash Memory Write Buffer Clear Control
0: Do not initiate Write Buffer Clear process or Write Buffer Clear process is completed
1: Initiate Write Buffer Clear process
This bit is set by software and cleared by hardware when the Write Buffer Clear process is completed.

- **FARL Register**

Bit	7	6	5	4	3	2	1	0
Name	A7	A6	A5	A4	A3	A2	A1	A0
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 Flash Memory Address bit 7~bit 0

- **FARH Register – BH66F2650**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	A12	A11	A10	A9	A8
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 Flash Memory Address bit 12~bit 8

- **FARH Register – BH66F2660**

Bit	7	6	5	4	3	2	1	0
Name	—	—	A13	A12	A11	A10	A9	A8
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 Flash Memory Address bit 13~bit 8

- **FD0L Register**

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

Bit 7~0 The first Flash Memory Data bit 7~bit 0

- **FD0H Register**

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

Bit 7~0 The first Flash Memory Data bit 15~bit 8

- **FD1L Register**

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

Bit 7~0 The second Flash Memory Data bit 7~bit 0

- **FD1H Register**

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

Bit 7~0 The second Flash Memory Data bit 15~bit 8

- **FD2L Register**

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

Bit 7~0 The third Flash Memory Data bit 7~bit 0

- **FD2H Register**

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

Bit 7~0 The third Flash Memory Data bit 15~bit 8

- **FD3L Register**

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

Bit 7~0 The fourth Flash Memory Data bit 7~bit 0

- **FD3H Register**

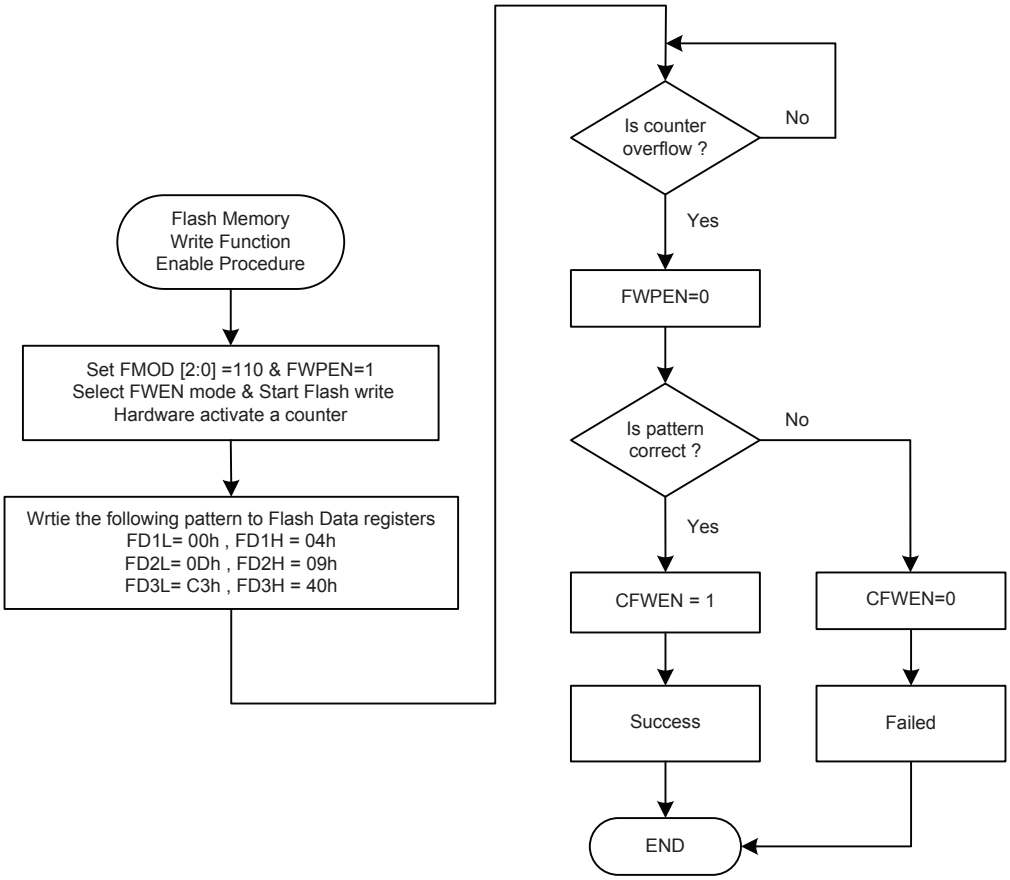
Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 The fourth Flash Memory Data bit 15~bit 8

Flash Memory Write Function Enable Procedure

In order to allow users to change the Flash memory data through the IAP control registers, users must first enable the Flash memory write operation by the following procedure:

1. Write “110” into the FMOD2~FMOD0 bits to select the FWEN mode.
2. Set the FWPEN bit to “1”. The step 1 and step 2 can be executed simultaneously.
3. The pattern data with a sequence of 00H, 04H, 0DH, 09H, C3H and 40H must be written into the FD1L, FD1H, FD2L, FD2H, FD3L and FD3H registers respectively.
4. A counter with a time-out period of 300μs will be activated to allow users writing the correct pattern data into the FD1L/FD1H~FD3L/FD3H register pairs. The counter clock is derived from LIRC oscillator.
5. If the counter overflows or the pattern data is incorrect, the Flash memory write operation will not be enabled and users must again repeat the above procedure. Then the FWPEN bit will automatically be cleared to “0” by hardware.
6. If the pattern data is correct before the counter overflows, the Flash memory write operation will be enabled and the FWPEN bit will automatically be cleared to “0” by hardware. The CFWEN bit will also be set to “1” by hardware to indicate that the Flash memory write operation is successfully enabled.
7. Once the Flash memory write operation is enabled, the user can change the Flash ROM data through the Flash control register.
8. To disable the Flash memory write operation, the user can clear the CFWEN bit to “0”.



Flash Memory Write Function Enable Procedure

Flash Memory Read/Write Procedure

After the Flash memory write function is successfully enabled through the preceding IAP procedure, users must first erase the corresponding Flash memory page and then initiate the Flash memory write operation. For the BH66F2650/BH66F2660 devices the number of the page erase operation are 32 words and 64 words per page respectively, the available page erase address is specified by FARH register and the content of FARL [7:5] and FARL[7:6]bit field respectively.

Erase Page	FARH	FARL [7:5]	FARL [4:0]
0	0000 0000	000	x xxxx
1	0000 0000	001	x xxxx
2	0000 0000	010	x xxxx
3	0000 0000	011	x xxxx
4	0000 0000	100	x xxxx
5	0000 0000	101	x xxxx
6	0000 0000	110	x xxxx
7	0000 0000	111	x xxxx
8	0000 0001	000	x xxxx
9	0000 0001	001	x xxxx
⋮	⋮	⋮	⋮
126	0000 1111	110	x xxxx
127	0000 1111	111	x xxxx
128	0001 0000	000	x xxxx
129	0001 0000	001	x xxxx
⋮	⋮	⋮	⋮
254	0001 1111	110	x xxxx
255	0001 1111	111	x xxxx

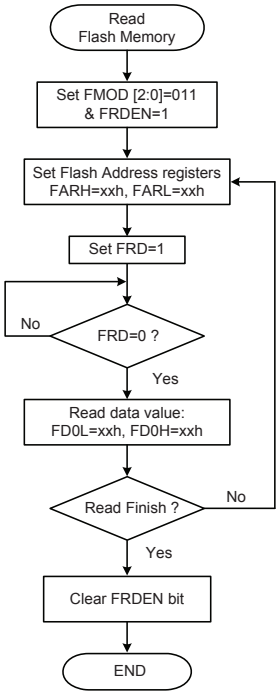
"x": don't care

BH66F2650 Erase Page Number and Selection

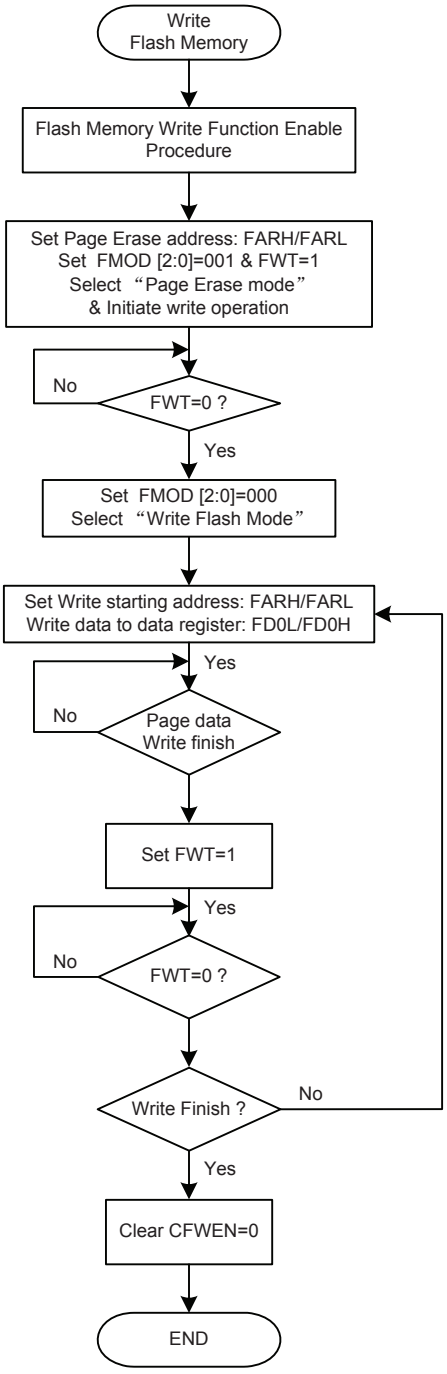
Erase Page	FARH	FARL [7:6]	FARL [5:0]
0	0000 0000	00	xx xxxx
1	0000 0000	01	xx xxxx
2	0000 0000	10	xx xxxx
3	0000 0000	11	xx xxxx
4	0000 0001	00	xx xxxx
5	0000 0001	01	xx xxxx
⋮	⋮	⋮	⋮
126	0001 1111	10	xx xxxx
127	0001 1111	11	xx xxxx
128	0010 0000	00	xx xxxx
129	0010 0000	01	xx xxxx
⋮	⋮	⋮	⋮
254	0011 1111	10	xx xxxx
255	0011 1111	11	xx xxxx

"x": don't care

BH66F2660 Erase Page Number and Selection



Read Flash Memory Procedure



Write Flash Memory Procedure

Note: When the FWT or FRD bit is set to "1", the MCU is stopped.

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.

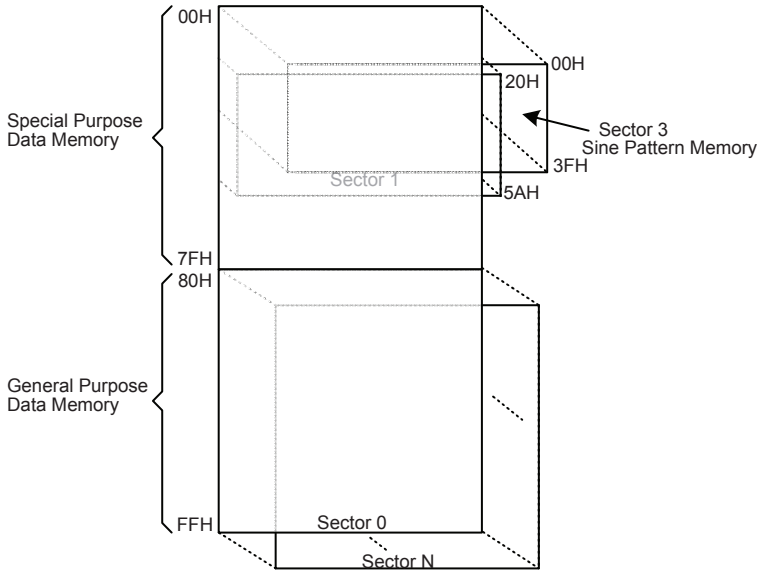
Categorised into three types, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the devices. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control. The third area is used for the Sine Pattern function. The addresses of the Sine Pattern Memory area overlop those in the Special Purpose Data Memory area.

Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide RAM. Each of the Data Memory Sector is categorized into three types, the special Purpose Data Memory and the General Purpose Data Memory. While the 00H~3FH of Sector 3 is Sine Pattern Memory.

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

Device	Special Purpose Data Memory	Sine Pattern Data Memory		General Purpose Data Memory	
	Available Sectors	Capacity	Address	Capacity	Address
BH66F2650	0: 00H~7FH 1: 20H~5AH	64×8	3: 00H~3FH	256×8	0: 80H~FFH 1: 80H~FFH
BH66F2660	0: 00H~7FH 1: 20H~5AH	64×8	3: 00H~3FH	1024×8	0: 80H~FFH 1: 80H~FFH : 6: 80H~FFH 7: 80H~FFH



Note: For the BH66F2650, N=1; For the BH66F2660, N=7

Data Memory Structure

Data Memory Addressing

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

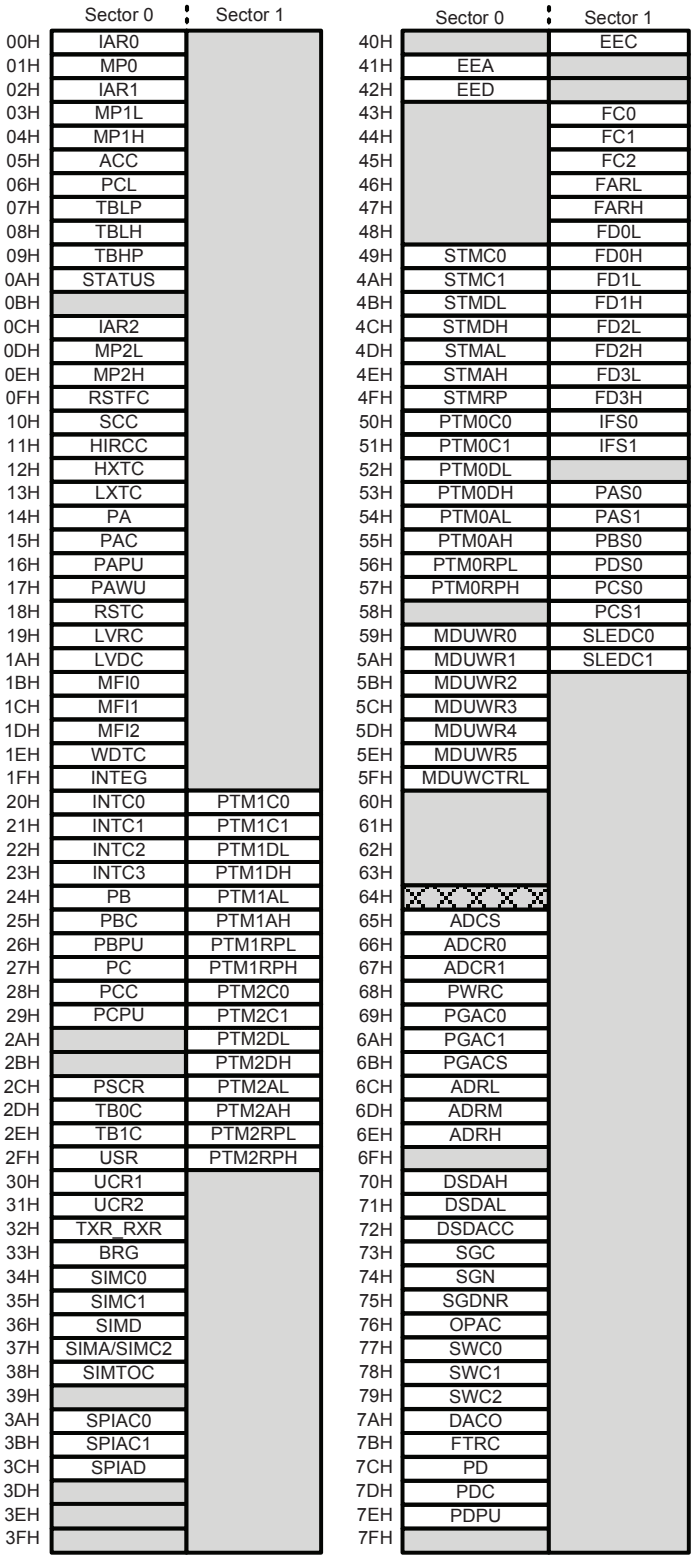
Direct Addressing can be used in all sectors using the corresponding instruction which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address "m" in the extended instructions has 9 or 11 valid bits depending on which device is selected, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

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

Special Purpose Data Memory

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



Special Purpose Data Memory Structure – BH66F2650

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH	PBP	
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	SCC	
11H	HIRCC	
12H	HXTC	
13H	LXTC	
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	RSTC	
19H	LVRC	
1AH	LVDC	
1BH	MFI0	
1CH	MFI1	
1DH	MFI2	
1EH	WDTC	
1FH	INTEG	
20H	INTC0	PTM1C0
21H	INTC1	PTM1C1
22H	INTC2	PTM1DL
23H	INTC3	PTM1DH
24H	PB	PTM1AL
25H	PBC	PTM1AH
26H	PBPU	PTM1RPL
27H	PC	PTM1RPH
28H	PCC	PTM2C0
29H	PCPU	PTM2C1
2AH		PTM2DL
2BH		PTM2DH
2CH	PSCR	PTM2AL
2DH	TB0C	PTM2AH
2EH	TB1C	PTM2RPL
2FH	USR	PTM2RPH
30H	UCR1	
31H	UCR2	
32H	TXR_RXR	
33H	BRG	
34H	SIMC0	
35H	SIMC1	
36H	SIMD	
37H	SIMA/SIMC2	
38H	SIMTOC	
39H		
3AH	SPIAC0	
3BH	SPIAC1	
3CH	SPIAD	
3DH		
3EH		
3FH		

	Sector 0	Sector 1
40H		EEC
41H	EEA	
42H	EED	
43H		FC0
44H		FC1
45H		FC2
46H		FARL
47H		FARH
48H		FD0L
49H	STMC0	FD0H
4AH	STMC1	FD1L
4BH	STMDL	FD1H
4CH	STMDH	FD2L
4DH	STMAL	FD2H
4EH	STMAH	FD3L
4FH	STMRP	FD3H
50H	PTM0C0	IFS0
51H	PTM0C1	IFS1
52H	PTM0DL	
53H	PTM0DH	PAS0
54H	PTM0AL	PAS1
55H	PTM0AH	PBS0
56H	PTM0RPL	PDS0
57H	PTM0RPH	PCS0
58H		PCS1
59H	MDUWR0	SLEDC0
5AH	MDUWR1	SLEDC1
5BH	MDUWR2	
5CH	MDUWR3	
5DH	MDUWR4	
5EH	MDUWR5	
5FH	MDUWCTRL	
60H		
61H		
62H		
63H		
64H		
65H	ADCS	
66H	ADCR0	
67H	ADCR1	
68H	PWRC	
69H	PGAC0	
6AH	PGAC1	
6BH	PGACS	
6CH	ADRL	
6DH	ADRM	
6EH	ADRH	
6FH		
70H	DSDAH	
71H	DSDAL	
72H	DSDACC	
73H	SGC	
74H	SGN	
75H	SGDNR	
76H	OPAC	
77H	SWC0	
78H	SWC1	
79H	SWC2	
7AH	DACO	
7BH	FTRC	
7CH	PD	
7DH	PDC	
7EH	PDPU	
7FH		

: Unused, read as 00H
: Reserved, cannot be changed

Special Purpose Data Memory Structure – BH66F2660

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, IAR2

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

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

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

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

Indirect Addressing Program Example 1

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

Indirect Addressing Program Example 2

```
data .section ´data´
adres1   db ?
adres2   db ?
adres3   db ?
adres4   db ?
block    db ?
code .section at 0 ´code´
org 00h
start:
     mov a, 04h              ; setup size of block
     mov block, a
     mov a, 01h              ; setup the memory sector
     mov mp1h, a
     mov a, offset adres1    ; Accumulator loaded with first RAM address
     mov mp1l, a             ; setup memory pointer with first RAM address
loop:
     clr IAR1                ; clear the data at address defined by MP1L
     inc mp1l                ; increment memory pointer MP1L
     sdz block               ; check if last memory location has been cleared
     jmp loop
continue:
```

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

Direct Addressing Program Example using extended instructions

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

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

Program Memory Bank Pointer – PBP

For the BH66F2660 device the Program Memory is divided into several banks. Selecting the required Program Memory area is achieved using the Program Memory Bank Pointer, PBP. The PBP register should be properly configured before the device executes the "Branch" operation using the "JMP" or "CALL" instruction. After that a jump to a non-consecutive Program Memory address which is located in a certain bank selected by the program memory bank pointer bits will occur.

PBP Register – BH66F2660

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **PBP0**: Select Program 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 SC flag, CZ flag, zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

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

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

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

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

STATUS Register

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

"x" unknown

Bit 7 **SC**: XOR Operation Result - performed by the OV flag and the MSB of the instruction operation result.

Bit 6 **CZ**: Operational Result of Different Flags for Different Instructions.

For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.

For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation zero flag.

For other instructions, the CZ flag will not be affected.

Bit 5 **TO**: Watchdog Time-Out Flag
0: After power up or executing the "CLR WDT" or "HALT" instruction
1: A watchdog time-out occurred.

Bit 4 **PDF**: Power Down Flag
0: After power up or executing the "CLR WDT" instruction
1: By executing the "HALT" instruction

Bit 3 **OV**: Overflow Flag
0: No overflow
1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa.

Bit 2 **Z**: Zero Flag
0: The result of an arithmetic or logical operation is not zero
1: The result of an arithmetic or logical operation is zero

Bit 1 **AC**: Auxiliary flag
0: No auxiliary carry
1: An operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction

Bit 0 **C**: Carry Flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

C is also affected by a rotate through carry instruction.

EEPROM Data Memory

These devices contain an area of internal EEPROM Data Memory. EEPROM, which stands for Electrically Erasable Programmable Read Only Memory, 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 64×8 ~ 256×8 bits for these devices. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and a data register in Sector 0 and a single control register in Sector 1.

EEPROM Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEA (BH66F2650)	—	—	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EEA (BH66F2660)	EEA7	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM Register List

- **EEA Register – BH66F2650**

Bit	7	6	5	4	3	2	1	0
Name	—	—	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **EEA5~EEA0**: Data EEPROM Address
Data EEPROM address bit 5~bit 0

- **EEA Register – BH66F2660**

Bit	7	6	5	4	3	2	1	0
Name	EEA7	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
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 **EEA7~EEA0**: Data EEPROM Address
Data EEPROM address bit 7~bit 0

- **EED Register**

Bit	7	6	5	4	3	2	1	0
Name	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
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 **EED7~EED0**: 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 "0" 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 "0" 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 "0" 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 "0" 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: 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.

Reading Data from the EEPROM

To read data from the EEPROM, the read enable bit, RDEN, in the EEC register must first be set high to enable the read function. The EEPROM address of the data to be read must then be placed in the EEA register. 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 "0", 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

The EEPROM address of the data to be written must first be placed in the EEA register and the data placed in the EED register. To write data to the EEPROM, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed consecutively. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set again after the write cycle has started. Note that setting the WR bit high will not initiate a write cycle if the WREN bit has not been set. As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to "0" by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended.

Write Protection

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

EEPROM Interrupt

The EEPROM write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. 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 "0" when not writing. Also the Memory Pointer high byte register, MP1H or MP2H, could be normally cleared to "0" as this would inhibit access to Sector 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, 40H               ; setup memory pointer MP1L
MOV  MP1L, A              ; MP1L points to EEC register
MOV  A, 01H               ; setup memory pointer MP1H
MOV  MP1H, A
SET  IAR1.1               ; set RDEN bit, enable read operations
SET  IAR1.0               ; start Read Cycle - set RD bit
BACK:
SZ   IAR1.0               ; check for read cycle end
JMP  BACK
CLR  IAR1                 ; disable EEPROM read/write
CLR  MP1H
MOV  A, EED               ; move read data to register
MOV  READ_DATA, A
```

- **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 MP1L
MOV  MP1L, A              ; MP1L points to EEC register
MOV  A, 01H               ; setup memory pointer MP1H
MOV  MP1H, A
CLR  EMI
SET  IAR1.3               ; set WREN bit, enable write operations
SET  IAR1.2               ; start Write Cycle - set WR bit - executed immediately
                          ; after set WREN bit
SET  EMI
BACK:
SZ   IAR1.2               ; check for write cycle end
JMP  BACK
CLR  IAR1                 ; disable EEPROM read/write
CLR  MP1H
```

Oscillators

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

Oscillator Overview

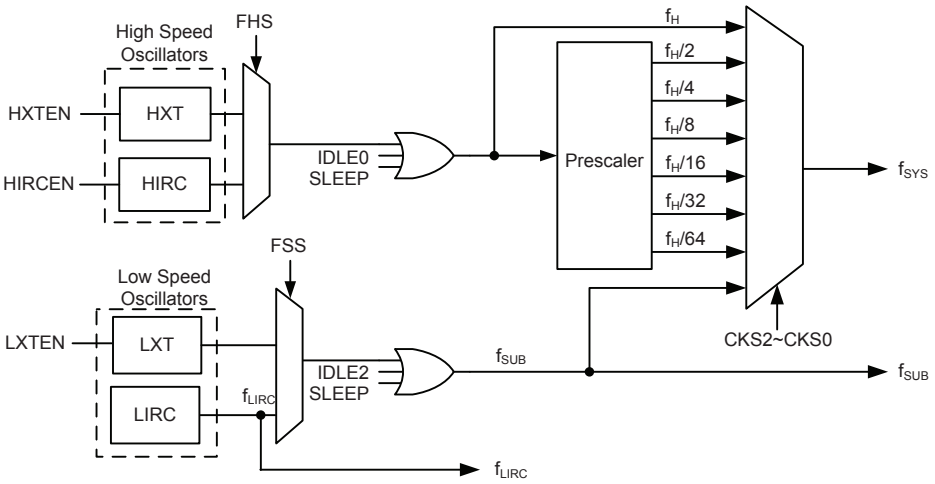
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided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

Type	Name	Freq.	Pins
External Crystal	HXT	4, 8, 12, 16MHz	OSC1/OSC2
Internal High Speed RC	HIRC	4, 8, 12MHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
Internal Low Speed RC	LIRC	32kHz	—

System Clock Configurations

There are four methods of generating the system clock, two high speed oscillators and two low speed oscillators. The high speed oscillators are the external crystal/ceramic oscillator and the internal 4/8/12MHz RC oscillator. The two low speed oscillators are the internal 32kHz RC oscillator and the external 32.768kHz crystal oscillator. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

The actual source clock used for the low speed oscillators is chosen via the FSS bit in the SCC register while for the high speed oscillator the source clock is selected by the FHS bit in the SCC register. The frequency of the slow speed or high speed system clock is determined using the CKS2~CKS0 bits in the SCC register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.

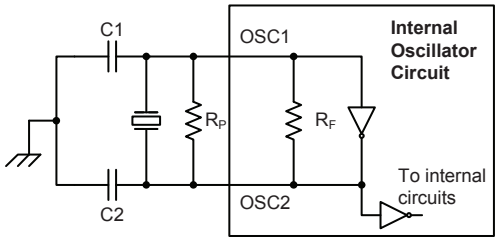


System Clock Configurations

External High Speed Crystal Oscillator − HXT

The simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation. However, for some crystals and most resonator types, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.

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



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

Crystal/Resonator Oscillator – HXT

HXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
12MHz	0pF	0pF
8MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

Internal RC Oscillator − HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 4/8/12MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins are free for use as normal I/O pins.

External 32.768kHz Crystal Oscillator − LXT

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

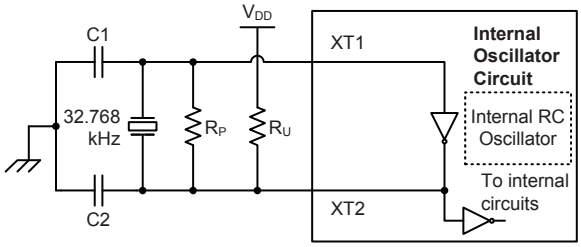
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE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

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

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

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

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



Note: 1. R_P, R_U, C1 and C2 are required.
2. Although not shown XT1/XT2 pins have a parasitic capacitance of around 7pF.

External LXT Oscillator

LXT Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
32.768kHz	10pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. R_P=5M~10MΩ is recommended. 3. R_U=10MΩ is recommended.		

32.768kHz Crystal Recommended Capacitor Values

Internal 32kHz Oscillator − LIRC

The internal 32kHz system oscillator is one of the low frequency oscillator choices, which is selected via a software control bit, FSS. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

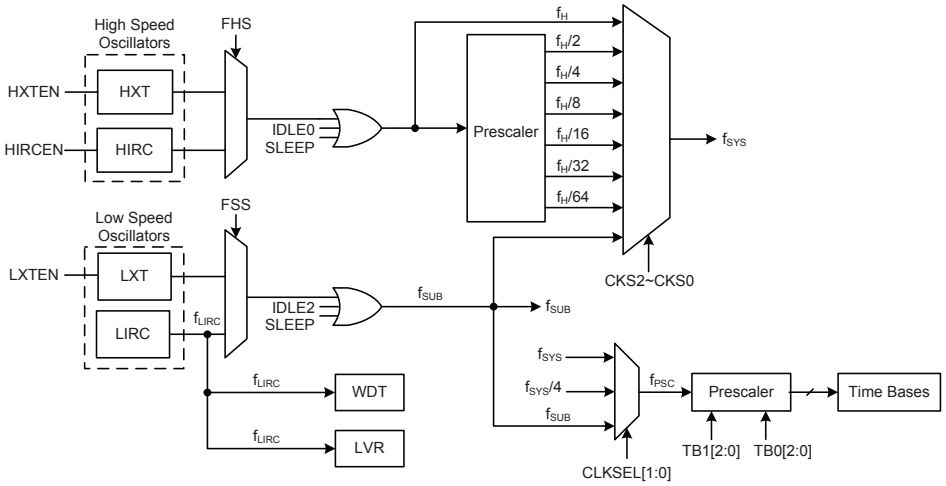
Operating Modes and System Clocks

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

System Clocks

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

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



Device Clock Configurations

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

System Operation Modes

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

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

"x" don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.

2. The f_{LIRC} clock can be switched on or off which is controlled by the WDT function being enabled or disabled.

FAST Mode

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

SLOW Mode

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

SLEEP Mode

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

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

The registers, SCC, HIRCC, HXTC and LXTC, are used for the overall control of the system clock within these devices.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	FHS	FSS	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
HXTC	—	—	—	—	—	HXTM	HXTF	HXTEN
LXTC	—	—	—	—	—	—	LXTF	LXTEN

System Operating Mode Control Registers List

SCC Register

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

Bit 7~5 **CKS2~CKS0**: System Clock Selection
000: f_H
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: f_{SUB}
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from f_H or f_{SUB}, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as "0"

Bit 3 **FHS**: High Frequency Clock Selection
0: HIRC
1: HXT

Bit 2 **FSS**: Low Frequency Clock Selection
0: LIRC
1: LXT

Bit 1 **FHIDEN**: High Frequency Oscillator Control when CPU is Switched Off
0: Disable
1: Enable
This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing an "HALT" instruction.

Bit 0 **FSIDEN**: Low Frequency Oscillator Control when CPU is Switched Off
0: Disable
1: Enable
This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing an "HALT" instruction. The LIRC oscillator is controlled by this bit together with the WDT function enable control when the LIRC is selected to be the low speed oscillator clock source or the WDT function is enabled respectively. If this bit is cleared to "0" but the WDT function is enabled, the LIRC oscillator will also be enabled.

HIRCC Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC Frequency Selection
00: 4MHz
01: 8MHz
10: 12MHz
11: 4MHz
When the HIRC oscillator is enabled or the HIRC frequency selection is changed by application program, the clock frequency will automatically be changed after the HIRCF flag is set to "1".

Bit 1 **HIRCF**: HIRC Oscillator Stable Flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to "1" to enable the HIRC oscillator or the HIRC frequency selection is changed by application program, the HIRCF bit will first be cleared to "0" and then set to "1" after the HIRC oscillator is stable.

Bit 0 **HIRCEN**: HIRC Oscillator Enable Control
0: Disable
1: Enable

HXTC Register

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

Bit 7~3 Unimplemented, read as "0"

Bit 2 **HXTM**: HXT Mode Selection
0: HXT frequency ≤ 10MHz
1: HXT frequency >10MHz
This bit is used to select the HXT oscillator operating mode. Note that this bit must be properly configured before the HXT is enabled. When the HXTEN bit is set to "1" to enable the HXT oscillator, it is invalid to change the value of this bit.

Bit 1 **HXTF**: HXT Oscillator Stable Flag
0: HXT unstable
1: HXT stable
This bit is used to indicate whether the HXT oscillator is stable or not. When the HXTEN bit is set to "1" to enable the HXT oscillator, the HXTF bit will first be cleared to "0" and then set to "1" after the HXT oscillator is stable.

Bit 0 **HXTEN**: HXT Oscillator Enable Control
0: Disable
1: Enable

LXTC Register

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

Bit 7~2 Unimplemented, read as "0"

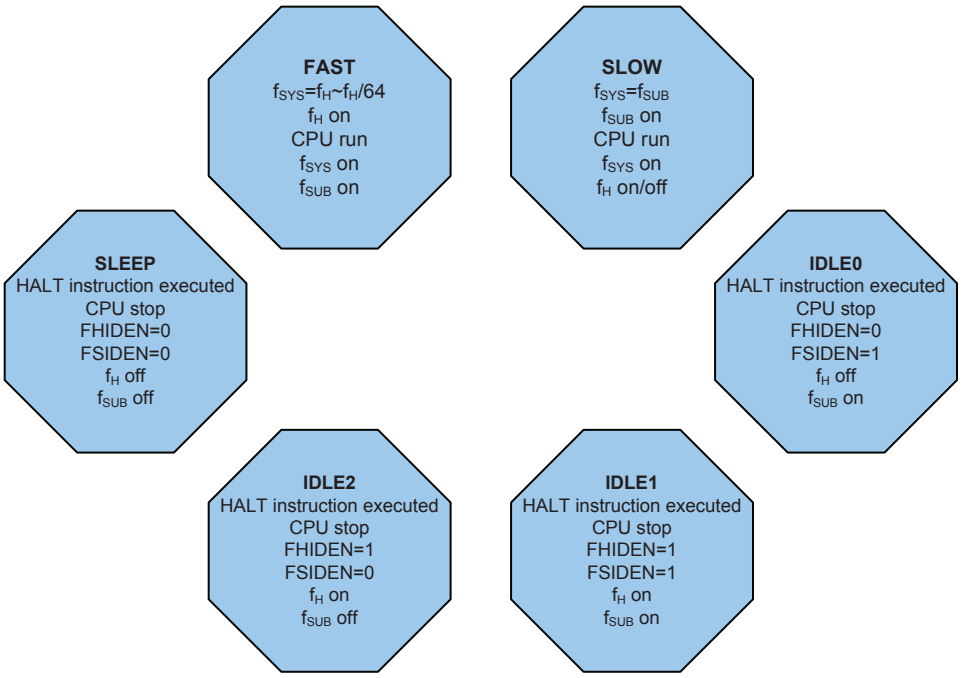
Bit 1 **LXTF**: LXT Oscillator Stable Flag
0: LXT unstable
1: LXT stable
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set to "1" to enable the LXT oscillator, the LXTF bit will first be cleared to "0" and then set to "1" after the LXT oscillator is stable.

Bit 0 **LXTEN**: LXT Oscillator Enable Control
0: Disable
1: Enable

Operating Mode Switching

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

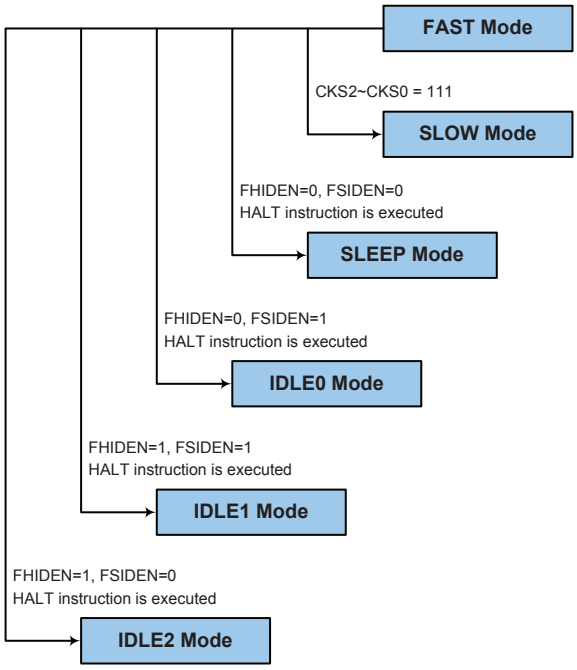
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When an HALT instruction is executed, whether the device enters the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



FAST Mode to SLOW Mode Switching

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

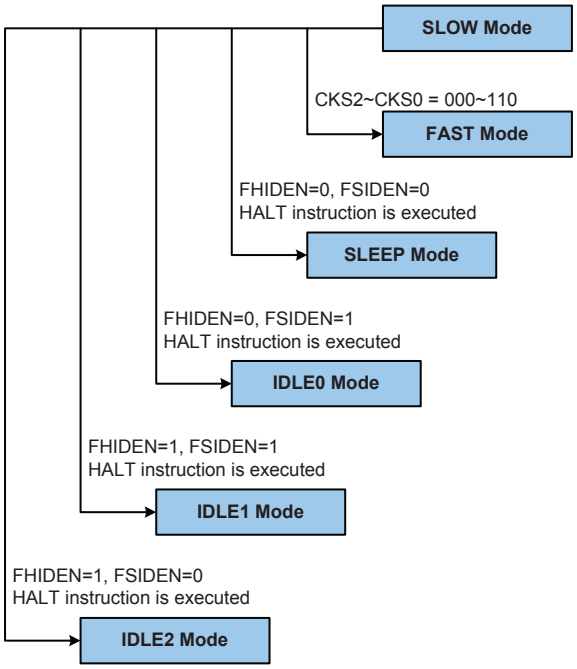
The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

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

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



Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

There is only one way for these devices to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

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

Entering the IDLE2 Mode

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

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

Standby Current Considerations

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

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

In the IDLE1 and IDLE 2 Mode the high speed oscillator is on, if the peripheral function clock source is derived from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

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

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

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

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

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{LIRC}, which is in turn supplied by the LIRC oscillator. 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. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations.

Watchdog Timer Control Register

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

WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT Function Software Control
10101: Disable
01010: Enable
Others: Reset MCU

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

Bit 2~0 **WS2~WS0**: WDT Time-out Period Selection
000: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$

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

RSTFC Register

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

"x" unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset Control Register Software Reset Flag
0: Not occurred
1: Occurred
This bit is set to "1" by the RSTC control register software reset and cleared by the application program. Note that this bit can only be cleared to "0" by the application program.

Bit 2 **LVRF**: LVR Function Reset Flag
Described elsewhere.

Bit 1 **LRF**: LVR Control Register Software Reset Flag
Described elsewhere.

Bit 0 **WRF**: WDT Control Register Software Reset Flag
0: Not occur
1: Occurred
This bit is set high by the WDT Control register software reset and cleared by the application program. Note that this bit can only be cleared to zero by the application program.

Watchdog Timer Operation

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

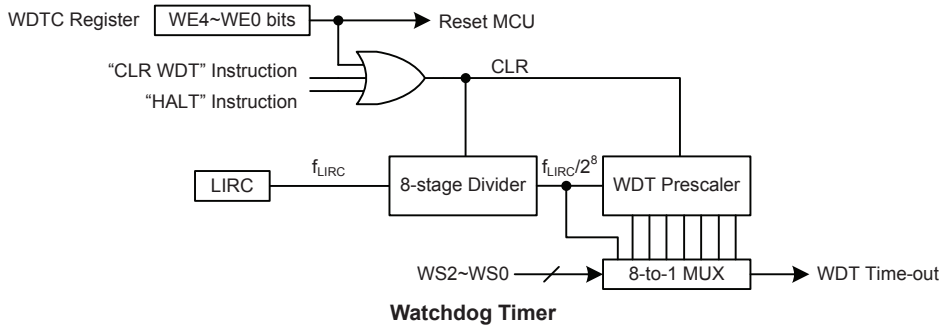
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other values	Reset MCU

Watchdog Timer Enable/Disable Control

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



Watchdog Timer

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontroller. In this case, internal circuitry will ensure that the microcontroller, after a short delay, will be in a well-defined state and ready to execute the first program instruction. After this power-on reset, certain important internal registers will be set to defined states before the program commences. One of these registers is the Program Counter, which will be reset to "0" forcing the microcontroller to begin program execution from the lowest Program Memory address.

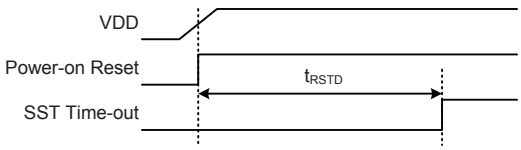
Another type of reset is when the Watchdog Timer overflows and resets. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, is implemented in situations where the power supply voltage falls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

Reset Functions

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

Power-on Reset

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



Power-On Reset Timing Chart

Internal Reset Control

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

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

Internal Reset Function Control

- **RSTC Register**

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

Bit 7~0 **RSTC7~RSTC0**: Reset Function Control
01010101: No operation
10101010: No operation
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET} and the RSTF bit in the RSTFC register will be set to "1".

- **RSTFC Register**

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

"x" unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset Control Register Software Reset Flag
0: Not occurred
1: Occurred
This bit is set to "1" by the RSTC control register software reset and cleared by the application program. Note that this bit can only be cleared to "0" by the application program.

Bit 2 **LVRF**: LVR Function Reset Flag
Described elsewhere.

Bit 1 **LRF**: LVR Control Register Software Reset Flag
Described elsewhere.

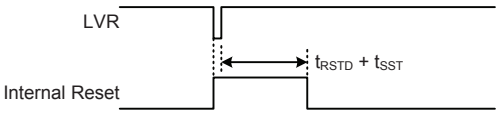
Bit 0 **WRF**: WDT Control Register Software Reset Flag
Described elsewhere.

In Application Programing Reset

The devices contain the IAP function. So there exists an IAP reset, which is caused by writing data 55H to FC1 register.

Low Voltage Reset – LVR

The microcontrollers contain a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVD/LVR characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise or software setting, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the power down mode.



Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR Voltage Select
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Other values: 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.

- **RSTFC Register**

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

"x" unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset Control Register Software Reset Flag
Described elsewhere.

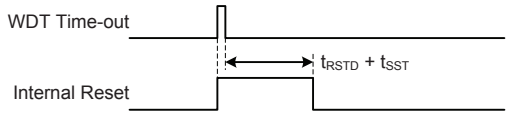
Bit 2 **LVRF**: LVR Function Reset Flag
0: Not occur
1: Occurred
This bit is set high when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to "0" by the application program.

Bit 1 **LRF**: LVR Control Register Software Reset Flag
0: Not occur
1: Occurred
This bit is set high if the LVRC register contains any non-defined LVR voltage register values. This in effect acts like a software-reset function. This bit can only be cleared to "0" by the application program.

Bit 0 **WRF**: WDT Control register software reset flag
Described elsewhere.

Watchdog Time-out Reset during Normal Operation

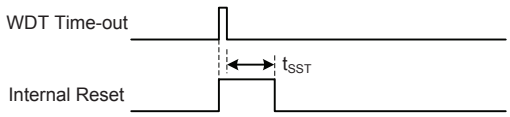
The Watchdog time-out Reset during normal operation is the same as LVR reset except that the Watchdog time-out flag TO will be set high.



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 high. Refer to the System Start Up Time Characteristics for t_{SST} details.



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

Reset Initial Conditions

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

TO	PDF	Reset Conditions
0	0	Power-on reset
u	u	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" stands for unchanged

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

Item	Condition after Reset
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time 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 Name	BH66F2650	BH66F2660	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	●	●	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	●	—	---x xxxx	---u uuuu	---u uuuu	---u uuuu
TBHP	—	●	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	●	●	xx00 xxxx	uuuu uuuu	xx1u uuuu	uu11 uuuu
PBP	—	●	---- ---0	---- ---0	---- ---0	---- ---u
IAR2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	●	●	---- 0x00	---- u1uu	---- uuuu	---- uuuu
SCC	●	●	000- 0000	000- 0000	000- 0000	uuu- uuuu
HIRCC	●	●	---- 0001	---- 0001	---- 0001	---- uuuu

Register Name	BH66F2650	BH66F2660	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
HXTC	●	●	---- -000	---- -000	---- -000	---- -uuu
LXTC	●	●	---- --00	---- --00	---- --00	---- --uu
PA	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTC	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
LVRC	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
LVDC	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
MFI0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI2	●	●	--00 --00	--00 --00	--00 --00	--uu --uu
WDTC	●	●	0101 0011	0101 0011	0101 0011	uuuu uuuu
INTEG	●	●	---- 0000	---- 0000	---- 0000	---- uuuu
INTC0	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	●	●	-000 -000	-000 -000	-000 -000	-uuu -uuu
INTC2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC3	●	●	---0 ---0	---0 ---0	---0 ---0	---u ---u
PB	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PSCR	●	●	---- --00	---- --00	---- --00	---- --uu
TB0C	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
TB1C	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
USR	●	●	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	●	●	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TXR_RXR	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMC0	●	●	111- 0000	111- 0000	111- 0000	uuu- uuuu
SIMC1	●	●	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMC2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMTOC	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SPIC0	●	●	111- --00	111- --00	111- --00	uuu- --uu
SPIC1	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
SPID	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
EEA	●	—	--00 0000	--00 0000	--00 0000	--uu uuuu
EEA	—	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
EED	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMC0	●	●	0000 0---	0000 0---	0000 0---	uuuu u---
STMC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register Name	BH66F2650	BH66F2660	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
STMDL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMRP	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0C0	●	●	0000 0---	0000 0---	0000 0---	uuuu u---
PTM0C1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DH	●	●	---- --00	---- --00	---- --00	---- --uu
PTM0AL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0AH	●	●	---- --00	---- --00	---- --00	---- --uu
PTM0RPL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	●	●	---- --00	---- --00	---- --00	---- --uu
MDUWR0	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR1	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR2	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR3	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR4	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR5	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWCTRL	●	●	00-- ----	00-- ----	00-- ----	uu-- ----
ADCS	●	●	---0 0000	---0 0000	---0 0000	---u uuuu
ADCR0	●	●	0010 00-0	0010 00-0	0010 00-0	uuuu uu-u
ADCR1	●	●	0000 000-	0000 000-	0000 000-	uuuu uuu-
PWRC	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
PGAC0	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
PGAC1	●	●	-000 000-	-000 000-	-000 000-	-uuu uuu-
PGACS	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADRL	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRM	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
DSDAH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
DSDAL	●	●	---- 0000	---- 0000	---- 0000	---- uuuu
DSDACC	●	●	000- ----	000- ----	000- ----	uuu- ----
SGC	●	●	0000 ----	0000 ----	0000 ----	uuuu ----
SGN	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
SGDNR	●	●	---0 0000	---0 0000	---0 0000	---u uuuu
OPAC	●	●	0--- 0000	0--- 0000	0--- 0000	u--- uuuu
SWC0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SWC1	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
SWC2	●	●	-000 000-	-000 000-	-000 000-	-uuu uuu-
DACO	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
FTRC	●	●	0--0 --00	0--0 --00	0--0 --00	u--u --uu
PD	●	●	---- 1111	---- 1111	---- 1111	---- uuuu
PDC	●	●	---- 1111	---- 1111	---- 1111	---- uuuu
PDPU	●	●	---- 0000	---- 0000	---- 0000	---- uuuu

Register Name	BH66F2650	BH66F2660	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PTM1C0	●	●	0000 0---	0000 0---	0000 0---	uuuu u---
PTM1C1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DH	●	●	---- --00	---- --00	---- --00	---- --uu
PTM1AL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1AH	●	●	---- --00	---- --00	---- --00	---- --uu
PTM1RPL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	●	●	---- --00	---- --00	---- --00	---- --uu
PTM2C0	●	●	0000 0---	0000 0---	0000 0---	uuuu u---
PTM2C1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DH	●	●	---- --00	---- --00	---- --00	---- --uu
PTM2AL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2AH	●	●	---- --00	---- --00	---- --00	---- --uu
PTM2RPL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	●	●	---- --00	---- --00	---- --00	---- --uu
EEC	●	●	---- 0000	---- 0000	---- 0000	---- uuuu
FC0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC2	●	●	---- ---0	---- ---0	---- ---0	---- ---u
FARL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	●	—	---0 0000	---0 0000	---0 0000	---u uuuu
FARH	—	●	--00 0000	--00 0000	--00 0000	--uu uuuu
FD0L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS0	●	●	00-- 0--0	00-- 0--0	00-- 0--0	uu-- u--u
IFS1	●	●	-000 --00	-000 --00	-000 --00	-uuu --uu
PAS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS0	●	●	0000 ----	0000 ----	0000 ----	uuuu ----
PCS0	●	●	--00 --00	--00 --00	--00 --00	--uu --uu
PCS1	●	●	---- 0000	---- 0000	---- 0000	---- uuuu
SLEDC0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC1	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu

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

Input/Output Ports

The 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.

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC5	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU4	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	—	—	—	—	PD3	PD2	PD1	PD0
PDC	—	—	—	—	PDC3	PDC2	PDC1	PDC0
PDPU	—	—	—	—	PDPU3	PDPU2	PDPU1	PDPU0

"—" unimplemented

I/O Logic Function Registers List

Pull-high Resistors

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

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

PxPU Register

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

PxPUn: I/O Port x Pin Pull-high Function Control

0: Disable

1: Enable

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

Port A Wake-up

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

Note that the wake-up function can be controlled by the wake-up control registers only when the pin-shared functional pin is selected as general purpose input/output and the MCU enters the Power down mode.

PAWU Register

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

PAWUn: 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~PDC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be 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	PxC5	PxC5	PxC4	PxC3	PxC2	PxC1	PxC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

PxCn: I/O Port x Pin Type Selection
0: Output
1: Input

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

I/O Port Source Current Control

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

SLEDC0 Register

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

Bit 7~6 **SLEDC07~SLEDC06**: PB7~PB4 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 5~4 **SLEDC05~SLEDC04**: PB3~PB0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 3~2 **SLEDC03~SLEDC02**: PA7~PA4 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 1~0 **SLEDC01~SLEDC00**: PA3~PA0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

SLEDC1 Register

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **SLEDC15~SLEDC14**: PD3~PD0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 3~2 **SLEDC13~SLEDC12**: PC7~PC4 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 1~0 **SLEDC11~SLEDC10**: PC3~PC0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

When the pin-shared input function is selected to be used, the corresponding input and output functions selection should be properly managed. For example, if the I^2C SDA line is used, the corresponding output pin-shared function should be configured as the SDI/SDA function by configuring the PxSn register and the SDA signal intput should be properly selected using the IFSn register. However, if the external interrupt function is selected to be used, the relevant output pin-shared function should be selected as an I/O function and the interrupt input signal should be selected.

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 INTn, xTCKn, xTPnI, etc, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bit fields. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be setup as 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
IFS0	PTP2IPS	PTP1IPS	—	—	PTCK2PS	—	—	STCKPS
IFS1	—	SCSAPS	SDIAPS	SCKAPS	—	—	RXPS1	RXPS0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PCS0	—	—	PCS05	PCS04	—	—	PCS01	PCS00
PCS1	—	—	—	—	PCS13	PCS12	PCS11	PCS10
PDS0	PDS07	PDS06	PDS05	PDS04	—	—	—	—

Pin-shared Function Selection Registers List

- **PAS0 Register**

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

Bit 7~6 **PAS07~PAS06**: PA3 Pin-Shared Function Selection
00: PA3/PTP1I
01: PTP1
10: SDOA
11: OSC2

Bit 5~4 **PAS05~PAS04**: PA2 Pin-Shared Function Selection
00: PA2/PTCK0
01: PA2/PTCK0
10: PA2/PTCK0
11: XT2

Bit 3~2 **PAS03~PAS02**: PA1 Pin-Shared Function Selection
00: PA1/INT0/STCK
01: PA1/INT0/STCK
10: LVDIN
11: PTP0B

Bit 1~0 **PAS01~PAS00**: PA0 Pin-Shared Function Selection
00: PA0/PTP0I
01: PA0/PTP0I
10: PTP0
11: XT1

- **PAS1 Register**

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

Bit 7~6 **PAS17~PAS16**: PA7 Pin-Shared Function Selection
00: PA7/PTCK2
01: PA7/PTCK2
10: PA7/PTCK2
11: SDI/SDA

Bit 5~4 **PAS15~PAS14**: PA6 Pin-Shared Function Selection
00: PA6/STCK
01: SDIA
10: PA6/STCK
11: RX

Bit 3~2 **PAS13~PAS12**: PA5 Pin-Shared Function Selection
00: PA5/STPI
01: STP
10: SCKA
11: TX

Bit 1~0 **PAS11~PAS10**: PA4 Pin-Shared Function Selection
00: PA4/PTP2I
01: PTP2
10: $\overline{\text{SCSA}}$
11: OSC1

- **PBS0 Register**

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

Bit 7~6 **PBS07~PBS06**: PB3 Pin-Shared Function Selection
00: PB3
01: RX
10: SDIA
11: SDO

Bit 5~4 **PBS05~PBS04**: PB2 Pin-Shared Function Selection
00: PB2
01: $\overline{\text{PB2}}$
10: $\overline{\text{SCS}}$
11: SCKA

Bit 3~2 **PBS03~PBS02**: PB1 Pin-Shared Function Selection
00: PB1/INT1
01: PB1/INT1
10: STPB
11: PB1/INT1

Bit 1~0 **PBS01~PBS00**: PB0 Pin-Shared Function Selection
00: PB0
01: PB0
10: SCK/SCL
11: PB0

- **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PCS05	PCS04	—	—	PCS01	PCS00
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PCS05~PCS04**: PC2 Pin-Shared Function Selection
00: PC2/PTP2I
01: PTP2
10: PC2/PTP2I
11: PC2/PTP2I

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **PCS01~PCS00**: PC0 Pin-Shared Function Selection
00: PC0/PTP1I
01: PC0/PTP1I
10: PTP1
11: PC0/PTP1I

- **PCS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PCS13	PCS12	PCS11	PCS10
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 **PCS13~PCS12**: PC5 Pin-Shared Function Selection
00: PC5
01: PC5
10: TX
11: SDOA

Bit 1~0 **PCS11~PCS10**: PC4 Pin-Shared Function Selection
00: PC4
01: PC4
10: PC4
11: $\overline{\text{SCSA}}$

- **PDS0 Register**

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

Bit 7~6 **PDS07~PDS06**: PD3 Pin-Shared Function Selection
00: PD3
01: PD3
10: TX
11: PD3

Bit 5~4 **PDS05~PDS04**: PD2 Pin-Shared Function Selection
00: PD2
01: PD2
10: RX
11: PD2

Bit 3~0 Unimplemented, read as "0"

- **IFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTP2IPS	PTP1IPS	—	—	PTCK2PS	—	—	STCKPS
R/W	R/W	R/W	—	—	R/W	—	—	R/W
POR	0	0	—	—	0	—	—	0

Bit 7 **PTP2IPS**: PTP2I Input Source Pin Selection
0: PTP2I on PA4
1: PTP2I on PC2

Bit 6 **PTP1IPS**: PTP1I Input Source Pin Selection
0: PTP1I on PA3
1: PTP1I on PC0

Bit 5~4 Unimplemented, read as "0"

Bit 3 **PTCK2PS**: PTCK2 Input Source Pin Selection
0: PTCK2 on PC3
1: PTCK2 on PA7

Bit 2~1 Unimplemented, read as "0"

Bit 0 **STCKPS**: STCK Input Source Pin Selection
0: STCK on PA1
1: STCK on PA6

- **IFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	SCSAPS	SDIAPS	SCKAPS	—	—	RXPS1	RXPS0
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 **SCSAPS**: $\overline{\text{SCSA}}$ Input Source Pin Selection
0: $\overline{\text{SCSA}}$ on PA4
1: $\overline{\text{SCSA}}$ on PC4

Bit 5 **SDIAPS**: SDIA Input Source Pin Selection
0: SDIA on PA6
1: SDIA on PB3

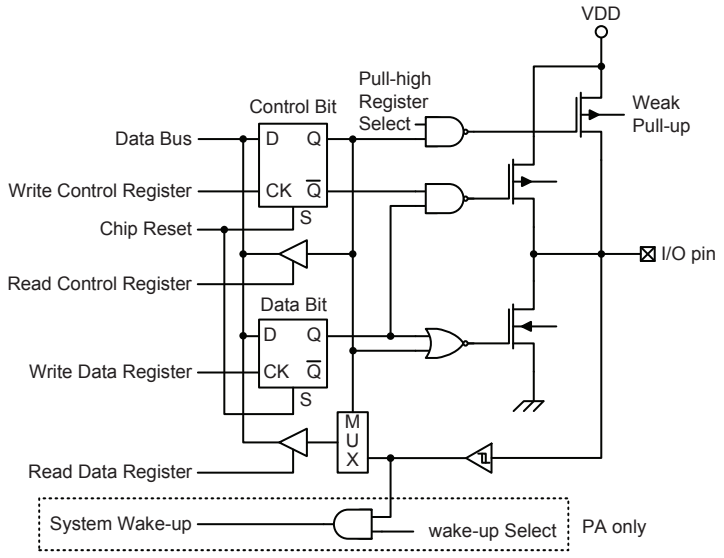
Bit 4 **SCKAPS**: SCKA Input Source Pin Selection
0: SCKA on PA5
1: SCKA on PB2

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **RXPS1~RXPS0**: RX Input Source Pin Selection
00: RX on PA6
01: RX on PB3
10: RX on PD2
11: RX on PD2

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.



Logic Function Input/Output Structure

Programming Considerations

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

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

Timer Modules – TM

One of the most fundamental functions in any microcontroller devices is the ability to control and measure time. To implement time related functions each 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 common features of the different TM types are described here with more detailed information provided in the individual Standard and Periodic TM sections.

Introduction

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

TM Function	STM	PTM
Timer/Counter	√	√
Input Capture	√	√
Compare Match Output	√	√
PWM Channels	1	1
Single Pulse Output	1	1
PWM Alignment	Edge	Edge
PWM Adjustment Period & Duty	Duty or Period	Duty or Period

TM Function Summary

Device	STM	PTM
BH66F2650	16-bit STM	10-bit PTM0 10-bit PTM1 10-bit PTM2
BH66F2660		

TM Name/Type Reference

TM Operation

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

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the STCK2~STCK0 and PTnCK2~PTnCK0 bits in the STM and PTMn 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 STCK and PTCKn pin. The STCK and PTCKn 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 or Periodic 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, irrespective of what type, has two TM input pins, with the label STCK, PTCKn and STPI, PTPnI respectively. The STM, PTMn input pin, STCK, PTCKn, is essentially a clock source for the STM, PTMn and is selected using the STCK2~STCK0 or PTnCK2~PTnCK0 bits in the STMC0 or PTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The STCK, PTCKn input pin can be chosen to have either a rising or falling active edge. The STCK and PTCKn pins are also used as the external trigger input pin in single pulse output mode for the STM and PTMn respectively.

The other STM, PTMn input pin, STPI, PTPnI, 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 STIO1~STIO0, PTnIO1~PTnIO0 bits in the STMC1, PTMnC1 register respectively. There is another capture input, PTCKn, for PTMn capture input mode, which can be used as the external trigger input source except the PTPnI pin.

The TMs each have one or more output pins. The STPB, PTP0B is the inverted signal of the STP, PTP0 output. The TM output pins 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 STP, PTPn or STPB, PTP0B output pin is also

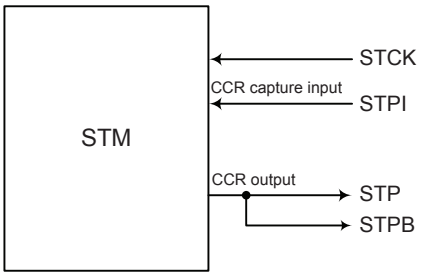
the pin where the TM generates the PWM output waveform. As the TM output pins are pin-shared with other functions, the TM output function must first be setup using relevant pin-shared function selection register.

STM		PTMn	
Input	Output	Input	Output
STCK, STPI	STP, STPB	PTCK0, PTP0I, PTCK1, PTP1I PTCK2, PTP2I	PTP0B, PTP0, PTP1, PTP2

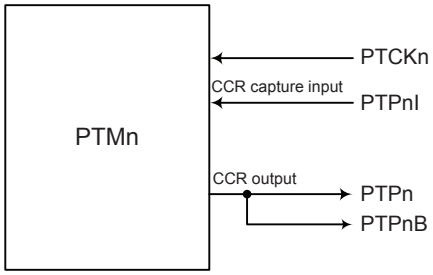
TM External Pins

TM Input/Output Pin Selection

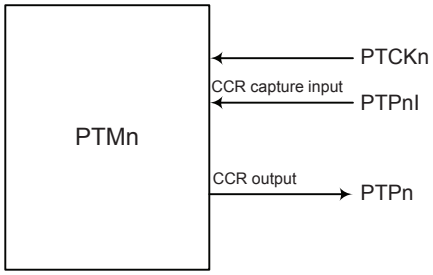
Selecting to have a TM input/output or whether to retain its other shared function is implemented using the relevant pin-shared function selection registers, with the corresponding selection bits in each pin-shared function register corresponding to a TM input/output pin. Configuring the selection bits correctly will setup the corresponding pin as a TM input/output. The details of the pin-shared function selection are described in the pin-shared function section.



STM Function Pin Control Block Diagram



PTM Function Pin Control Block Diagram – n=0

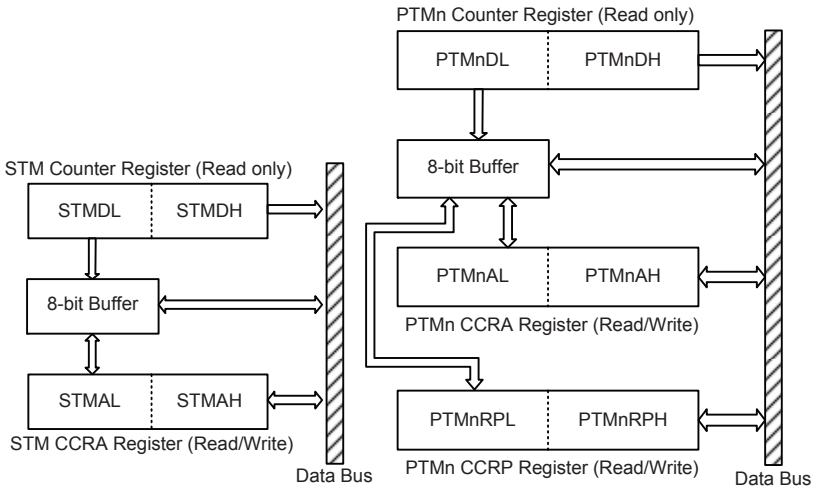


PTM Function Pin Control Block Diagram – n=1 or 2

Programming Considerations

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

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



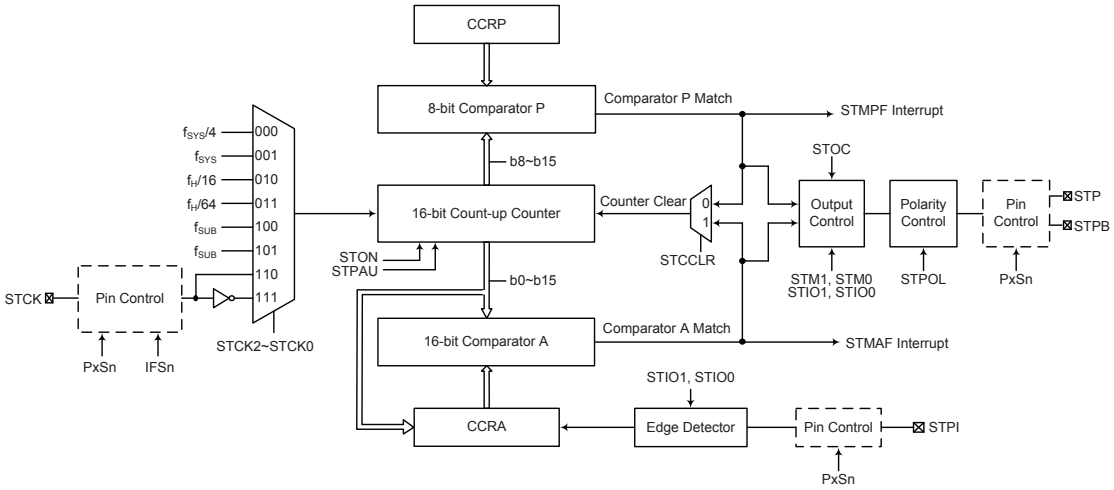
The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - Step 1. Write data to Low Byte STMAL, PTMnAL or PTMnRPL
    – note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte STMAH, PTMnAH or PTMnRPH
    – here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA or CCRP
  - Step 1. Read data from the High Byte STMDH, PTMnDH, STMAH, PTMnAH or PTMnRPH
    – here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - Step 2. Read data from the Low Byte STMDL, PTMnDL, STMAL, PTMnAL or PTMnRPL
    – this step reads data from the 8-bit buffer.

Standard Type TM – STM

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

STM Core	STM Input Pin	STM Output Pin
16-bit STM	STCK, STPI	STP, STPB



Standard Type TM Block Diagram

Standard TM Operation

The size of Standard TM is 16-bit wide and its core is a 16-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP comparator is 8-bit wide whose value is compared with the highest 8 bits in the counter while the CCRA is the sixteen bits and therefore compares all counter bits.

The only way of changing the value of the 16-bit counter using the application program, is to clear the counter by changing the STON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a STM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control 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 16-bit value, while a read/write register pair exists to store the internal 16-bit CCRA value. The STMRP register is used to store the 8-bit CCRP bits. The remaining two registers are control registers which setup the different operating and control modes.

Register Name	Bit							
	7	6	5	4	3	2	1	0
STMC0	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
STMC1	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
STMDL	D7	D6	D5	D4	D3	D2	D1	D0
STMDH	D15	D14	D13	D12	D11	D10	D9	D8
STMAL	D7	D6	D5	D4	D3	D2	D1	D0
STMAH	D15	D14	D13	D12	D11	D10	D9	D8
STMRP	D7	D6	D5	D4	D3	D2	D1	D0

16-bit Standard TM Registers List

STMDL Register

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

Bit 7~0 **D7~D0**: STM Counter Low Byte Register bit 7~bit 0
STM 16-bit Counter bit 7~bit 0

STMDH Register

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

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

STMAL Register

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

Bit 7~0 **D7~D0**: STM CCRA Low Byte Register bit 7~bit 0
STM 16-bit CCRA bit 7~bit 0

STMAH Register

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

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

STMC0 Register

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

Bit 7 **STPAU**: STM Counter Pause Control
0: Run
1: Pause

The counter can be paused by setting this bit high. Clearing the bit to "0" restores normal counter operation. When in a Pause condition the STM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **STCK2~STCK0**: Select STM Counter Clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: STCK rising edge clock
111: STCK falling edge clock

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

Bit 3 **STON**: STM Counter On/Off Control
0: Off
1: On

This bit controls the overall on/off function of the STM. Setting the bit high enables the counter to run while clearing the bit disables the STM. Clearing this bit to zero will stop the counter from counting and turn off the STM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the STM is in the Compare Match Output Mode or the PWM Output Mode or Single Pulse Output Mode then the STM output pin will be reset to its initial condition, as specified by the STOC bit, when the STON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

STMC1 Register

Bit	7	6	5	4	3	2	1	0
Name	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **STM1~STM0**: Select STM Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

Bit 5~4 **STIO1~STIO0**: Select STM External Pin (STP or STPI) 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 STPI
01: Input capture at falling edge of STPI
10: Input capture at rising/falling edge of STPI
11: Input capture disabled

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a compare match occurs from the Comparator A. The 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 "0", then no change will take place on the output. The initial value of the STM output pin should be setup using the STOC bit in the STMC1 register. Note that the output level requested by the STIO1 and STIO0 bits must be different from the initial value setup using the STOC bit otherwise no change will occur on the STM output pin when a compare match occurs. After the STM output pin changes state, it can be reset to its initial level by changing the level of the STON bit from low to high.

In the PWM Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the STIO1 and STIO0 bits only after the STM has been switched off. Unpredictable PWM outputs will occur if the STIO1 and STIO0 bits are changed when the STM is running.

Bit 3 **STOC**: STM STP Output Control

Compare Match Output Mode
0: Initial low
1: Initial high

PWM Output Mode/Single Pulse Output Mode
0: Active low
1: Active high

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

Bit 2 **STPOL**: STM STP Output Polarity Control
0: Non-inverted
1: Inverted

This bit controls the polarity of the STP output pin. When the bit is set high the STM output pin will be inverted and not inverted when the bit is "0". It has no effect if the STM is in the Timer/Counter Mode.

Bit 1 **STDPX**: STM PWM Duty/Period Control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

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

Bit 0 **STCCLR**: STM Counter Clear Condition Selection
0: Comparator P match
1: Comparator A match

This bit is used to select the method which clears the counter. Remember that the Standard TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the STCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to "0". The STCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

STMRP Register

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

Bit 7~0 **D7~D0**: STM CCRP 8-bit Register, Compared with the STM Counter bit 15~bit 8

Comparator P match period
0: 65536 STM clocks
1~255: (1~255) × 256 STM clocks

These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight bits. The result of this comparison can be selected to clear the internal counter if the STCCLR bit is set to "0". Setting the STCCLR bit to "0" ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to "0" is in effect allowing the counter to overflow at its maximum value.

Standard Type TM Operation Modes

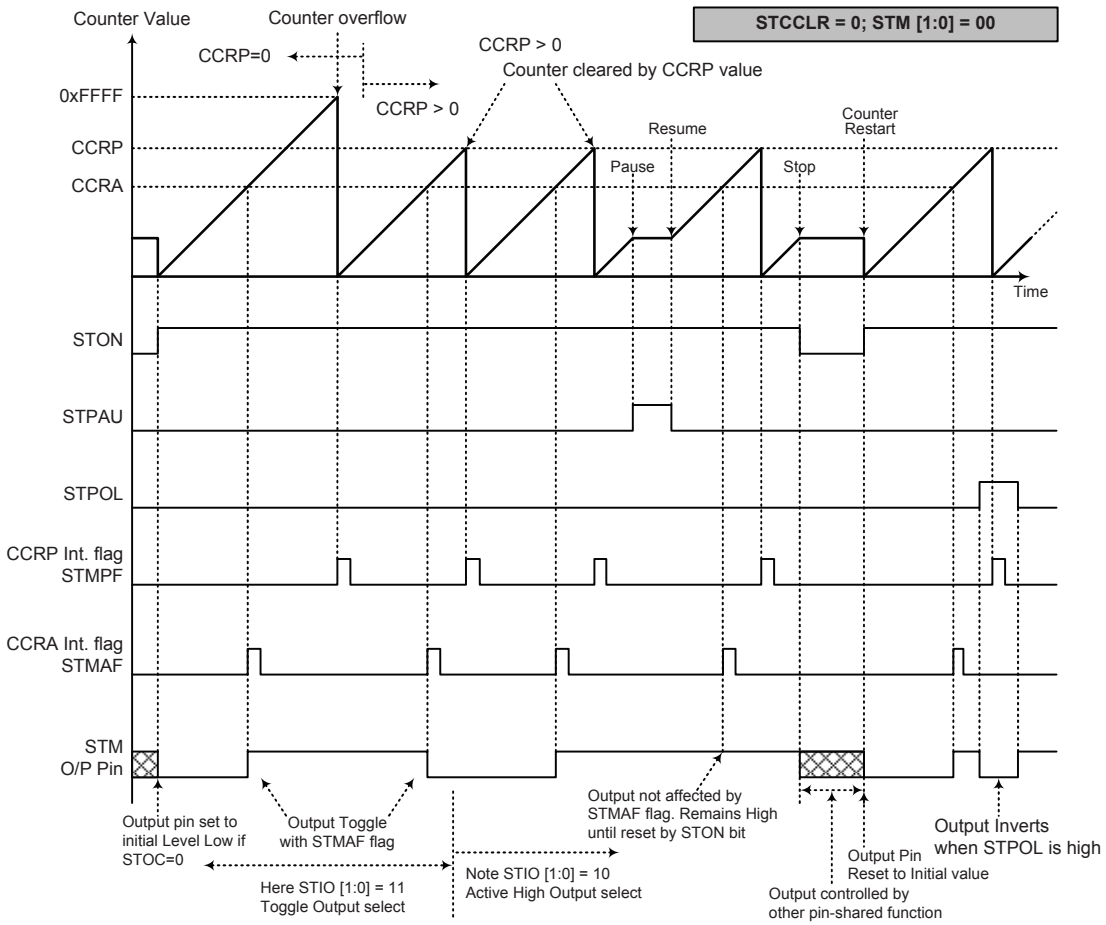
The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

Compare Match Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register, should be set to "00" respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the STCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all "0" which allows the counter to overflow. Here both STMAF and STMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

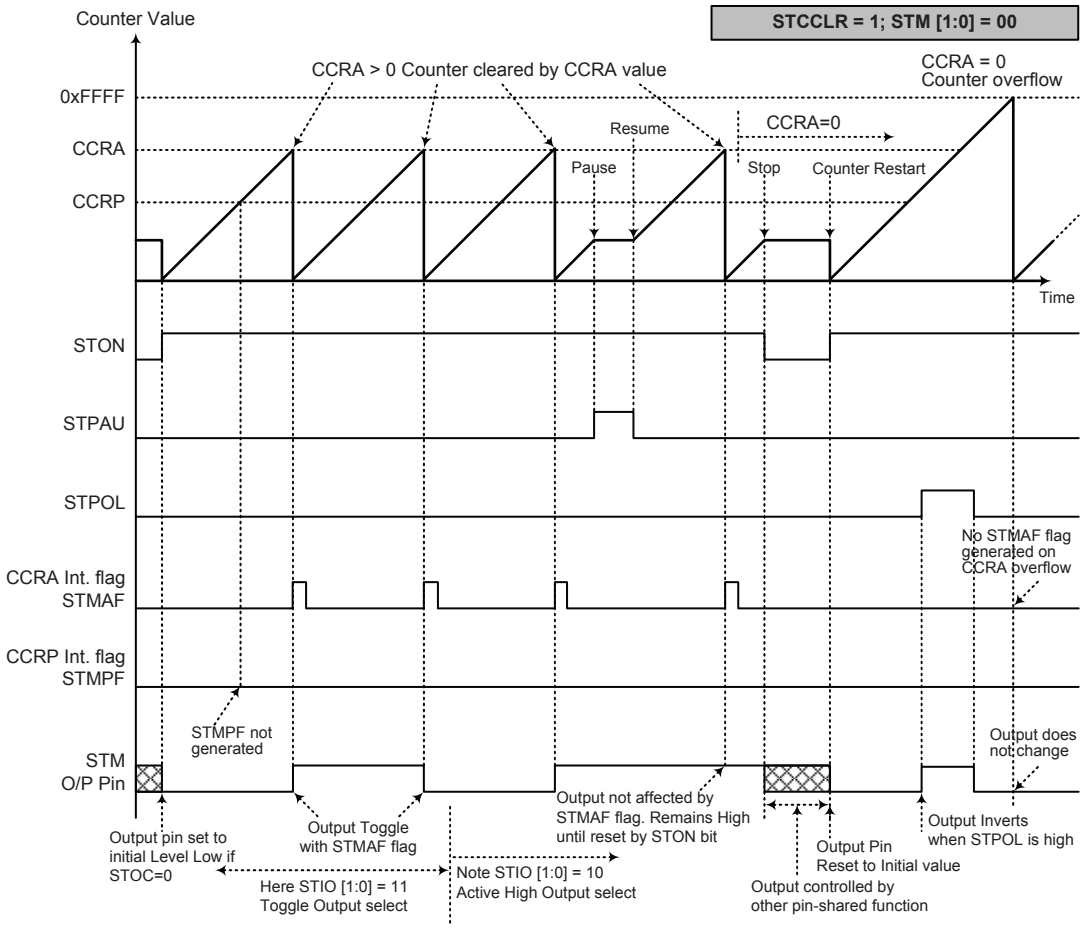
If the STCCLR bit in the STMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when STCCLR is high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

As the name of the mode suggests, after a comparison is made, the STM output pin, will change state. The STM output pin condition however only changes state when a STMAF interrupt request flag is generated after a compare match occurs from Comparator A. The STMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the STM output pin. The way in which the STM output pin changes state are determined by the condition of the STIO1 and STIO0 bits in the STMC1 register. The STM output pin can be selected using the STIO1 and STIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the STM output pin, which is setup after the STON bit changes from low to high, is setup using the STOC bit. Note that if the STIO1 and STIO0 bits are "0" then no pin change will take place.



Compare Match Output Mode – STCCLR=0

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



Compare Match Output Mode – STCCLR=1

Note: 1. With STCCLR=1 a Comparator A match will clear the counter

2. The STM output pin is controlled only by the STMAF flag

3. The output pin is reset to its initial state by a STON bit rising edge

4. A STMPF flag is not generated when STCCLR=1

Timer/Counter Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to "11" respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the STM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the STM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to "10" respectively and also the STIO1 and STIO0 bits should be set to "10" respectively. The PWM function within the STM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the STM output pin, a square 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 mode, the STCCLR 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 STDPX bit in the STMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The STOC bit in the STMC1 register is used to select the required polarity of the PWM waveform while the two STIO1 and STIO0 bits are used to enable the PWM output or to force the STM output pin to a fixed high or low level. The STPOL bit is used to reverse the polarity of the PWM output waveform.

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

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

If f_{SYS} = 16MHz, TM clock source is f_{SYS}/4, CCRP = 2 and CCRA = 128,

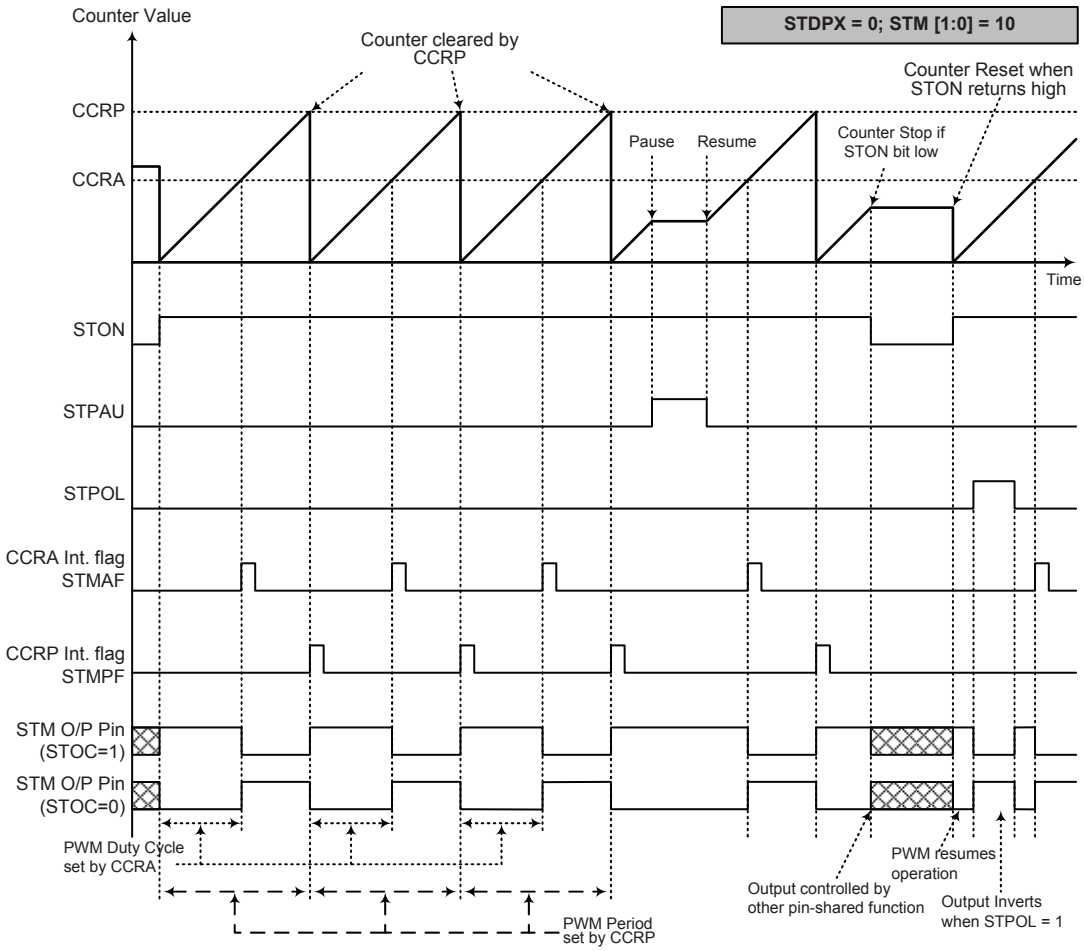
The STM PWM output frequency = (f_{SYS}/4) / (2×256) = f_{SYS} / 2048 = 7.8125 kHz,
duty = 128 / (2×256) = 25%.

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

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

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

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



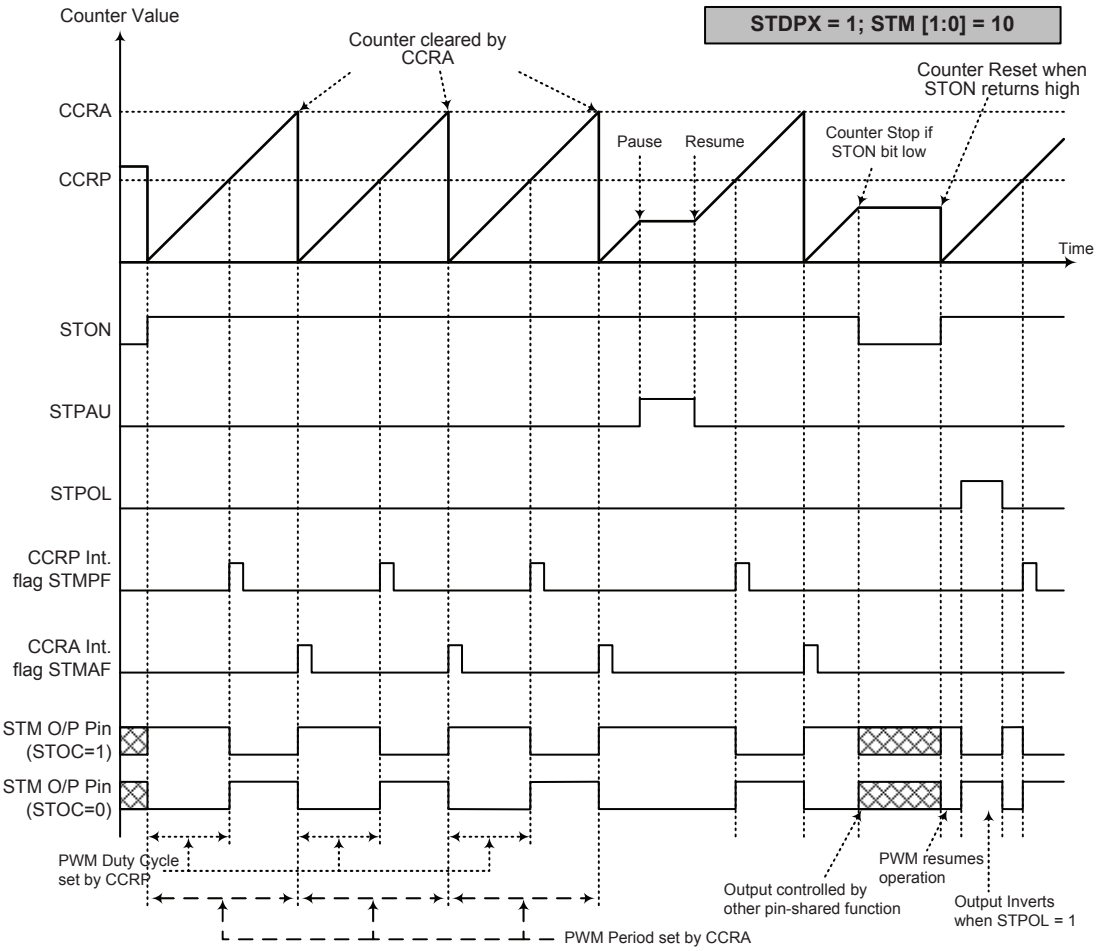
PWM Output Mode – STDPX=0

Note: 1. Here STDPX=0 – Counter cleared by CCRP

2. A counter clear sets the PWM Period

3. The internal PWM function continues running even when STIO [1:0]=00 or 01

4. The STCCLR bit has no influence on PWM operation



PWM Output Mode – STDPX=1

Note: 1. Here STDPX=1 – Counter cleared by CCRA

2. A counter clear sets the PWM Period

3. The internal PWM function continues even when STIO [1:0]=00 or 01

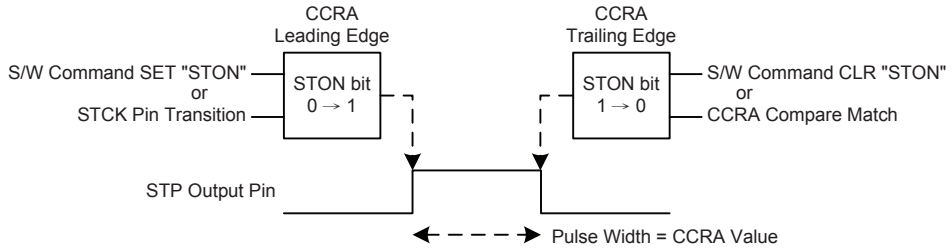
4. The STCCLR bit has no influence on PWM operation

Single Pulse Output Mode

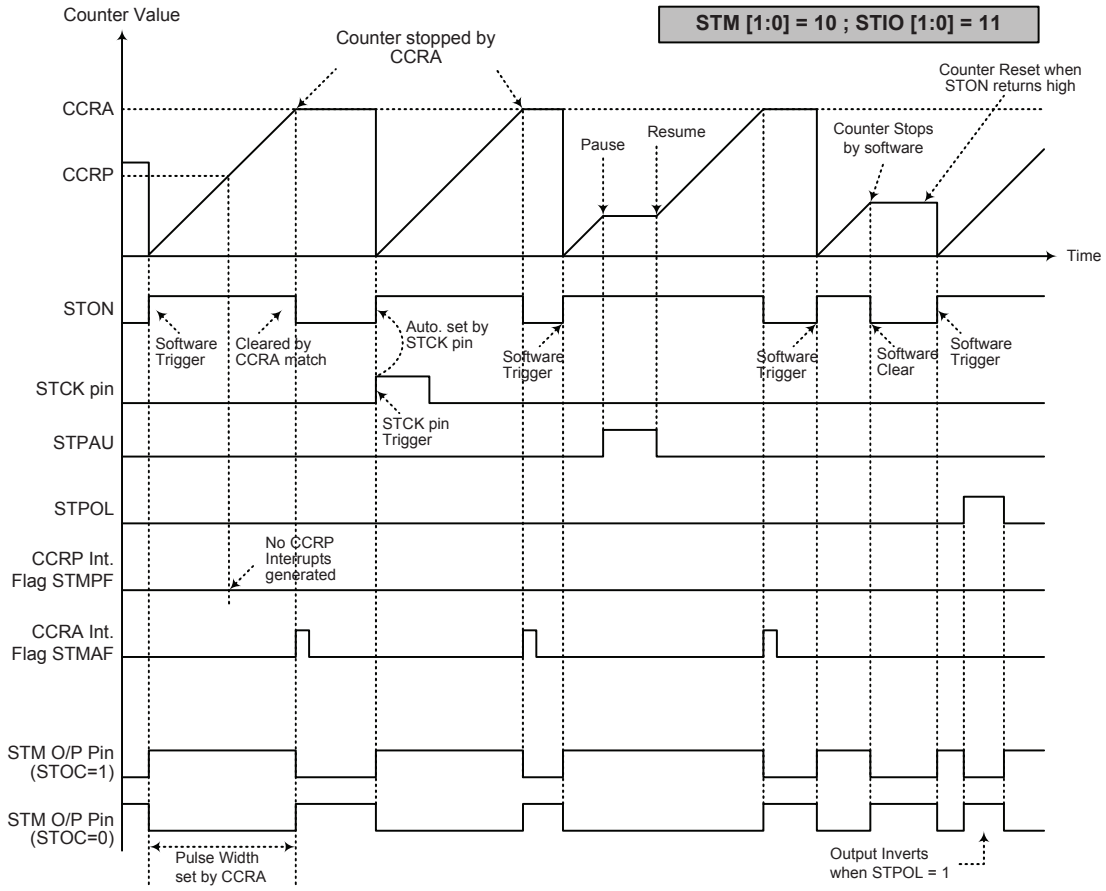
To select this mode, bits STM1 and STM0 in the STMC1 register should be set to "10" respectively and also the STIO1 and STIO0 bits should be set to "11" respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

The trigger for the pulse output leading edge is a low to high transition of the STON bit, which can be implemented using the application program. However in the Single Pulse Mode, the STON bit can also be made to automatically change from low to high using the external STCK pin, which will in turn initiate the Single Pulse output. When the STON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The STON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the STON bit is cleared to "0", which can be implemented using the application program or when a compare match occurs from Comparator A.

However a compare match from Comparator A will also automatically clear the STON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate a STM interrupt. The counter can only be reset back to "0" when the STON bit changes from low to high when the counter restarts. In the Single Pulse Mode CCRP is not used. The STCCLR and STDPX bits are not used in this Mode.



Single Pulse Generation



Single Pulse Mode

Note: 1. Counter stopped by CCRA

2. CCRP is not used

3. The pulse triggered by the STCK pin or by setting the STON bit high

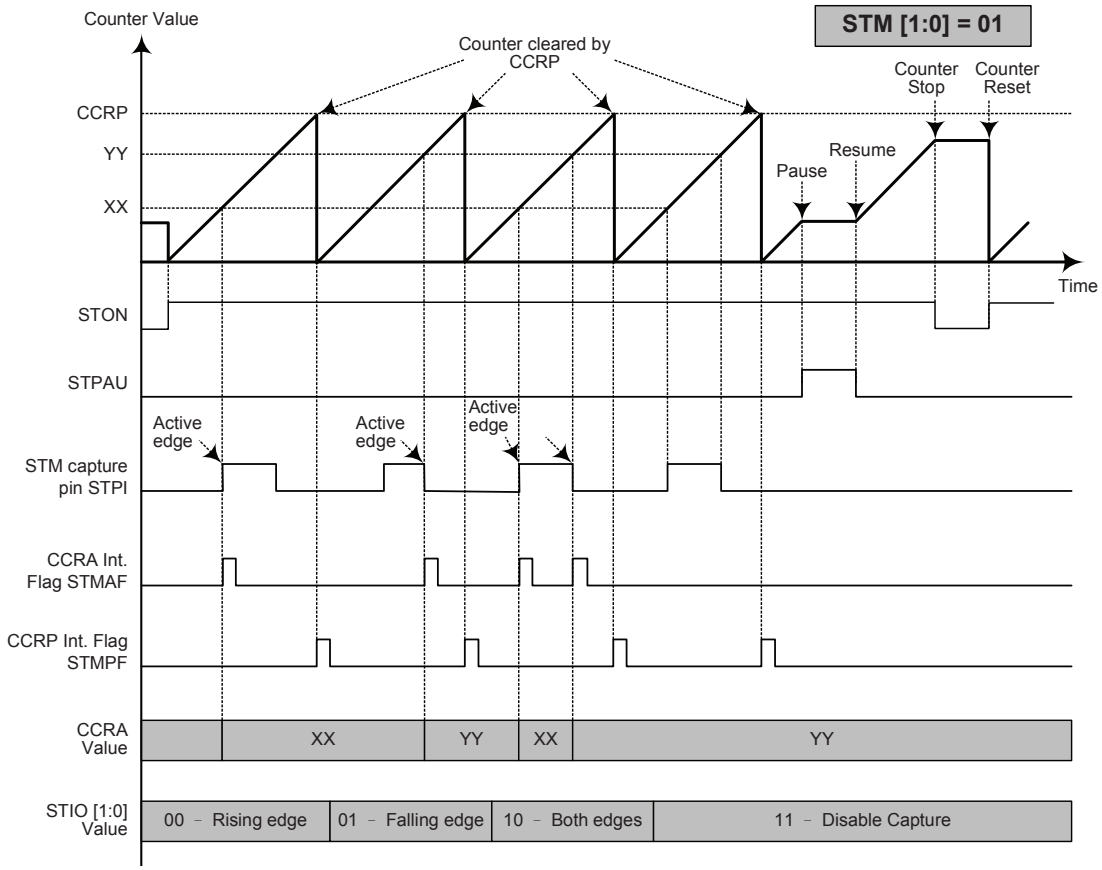
4. A STCK pin active edge will automatically set the STON bit high.

5. In the Single Pulse Mode, STIO [1:0] must be set to “11” and can not be changed.

Capture Input Mode

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

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

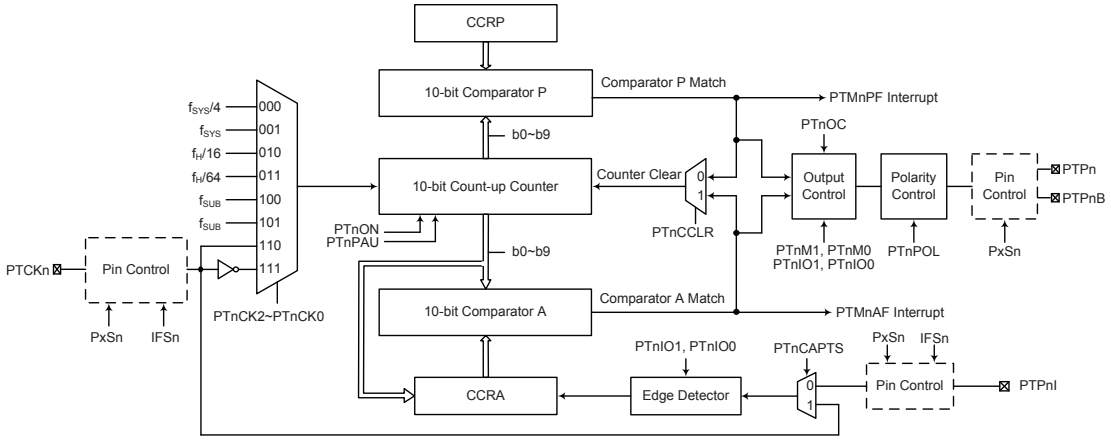


Capture Input Mode

Note: 1. STM [1:0]=01 and active edge set by the STIO [1:0] bits
2. A STM Capture input pin active edge transfers the counter value to CCRA
3. STCCLR bit not used
4. No output function − STOC and STPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to “0”.

Periodic Type TM – PTM

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



Note: The PTPnB pin only exists when n=0.

Periodic Type TM Block Diagram (n=0, 1 or 2)

Periodic TM Operation

The Periodic Type TM core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP comparator is 10-bit wide.

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

Periodic Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

10-bit Periodic TM Register List (n=0, 1 or 2)

PTMnC0 Register

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

Bit 7 **PTnPAU**: PTMn Counter Pause Control
0: Run
1: Pause

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

Bit 6~4 **PTnCK2~PTnCK0**: Select PTMn Counter Clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: PTCKn rising edge clock
111: PTCKn falling edge clock

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

Bit 3 **PTnON**: PTMn Counter On/Off Control
0: Off
1: On

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

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

Bit 2~0 Unimplemented, read as “0”

PTMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PTnM1~PTnM0**: Select PTMn Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

Bit 5~4 **PTnIO1~PTnIO0**: Select PTMn External Pin (PTPn or PTPnI/PTCKn) 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 PTPnI or PTCKn
01: Input capture at falling edge of PTPnI or PTCKn
10: Input capture at falling/rising edge of PTPnI or PTCKn
11: Input capture disabled

Timer/Counter Mode
Unused

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

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

In the PWM Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn 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 PTnIO1 and PTnIO0 bits only after the TM has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the PTMn is running.

Bit 3 **PTnOC**: PTMn PTPn Output Control bit

Compare Match Output Mode
0: Initial low
1: Initial high

PWM Output Mode/Single Pulse Output Mode
0: Active low
1: Active high

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

Bit 2 **PTnPOL**: PTMn PTPn Output Polarity Control
0: Non-invert
1: Invert

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

Bit 1 **PTnCAPTS**: PTMn Capture Trigger Source Selection
0: From PTPnI pin
1: From PTCKn pin

Bit 0 **PTnCCLR**: Select PTMn Counter Clear Condition
0: PTMn Comparator P match
1: PTMn Comparator A match

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

PTMnDL Register

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

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

PTMnDH Register

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

Bit 7~2 Unimplemented, read as "0"

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

PTMnAL Register

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

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

PTMnAH Register

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

Bit 7~2 Unimplemented, read as "0"

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

PTMnRPL Register

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

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

PTMnRPH Register

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

Bit 7~2 Unimplemented, read as "0"

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

Periodic Type TM Operating Modes

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

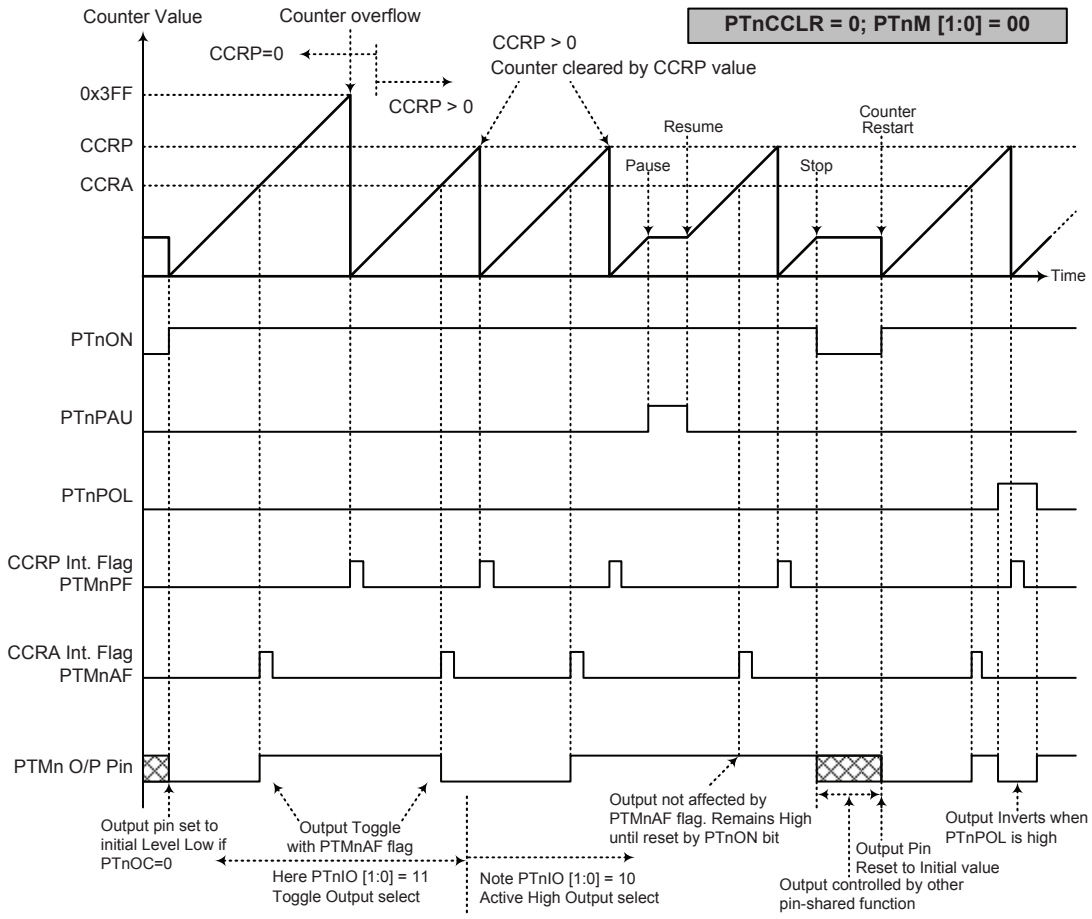
Compare Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register, should be set to "00" respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the PTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both PTMnAF and PTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

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

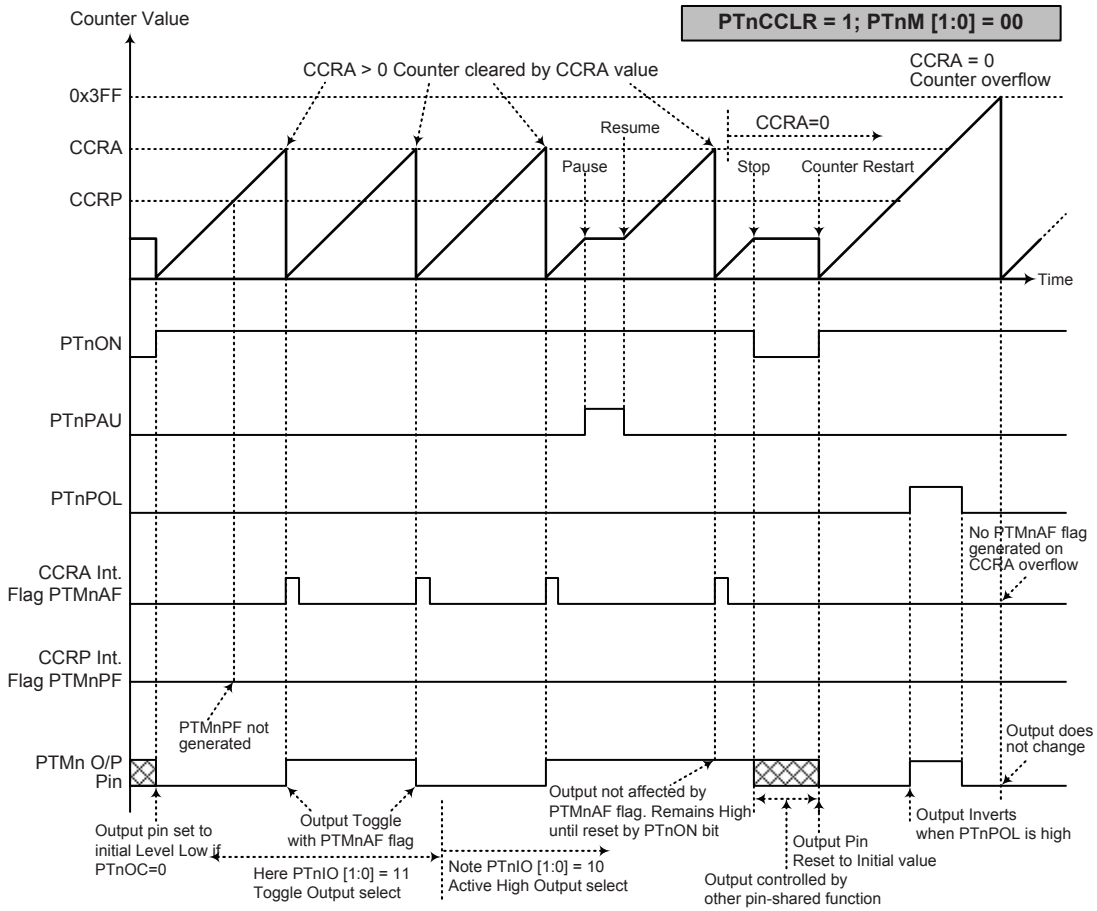
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the PTMnAF interrupt request flag will not be generated.

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



Compare Match Output Mode – PTnCCLR=0 (n=0, 1 or 2)

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



Compare Match Output Mode – PTnCCLR=1 (n=0, 1 or 2)

Note: 1. With PTnCCLR=1 a Comparator A match will clear the counter

2. The PTMn output pin is controlled only by the PTMnAF flag

3. The output pin is reset to its initial state by a PTnON bit rising edge

4. A PTMnPF flag is not generated when PTnCCLR=1

Timer/Counter Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to "11" respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the PTMn output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the PTMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to "10" respectively. The PWM function within the PTMn is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the PTMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

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

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The PTnOC bit in the PTMnC1 register is used to select the required polarity of the PWM waveform while the two PTnIO1 and PTnIO0 bits are used to enable the PWM output or to force the PTMn output pin to a fixed high or low level. The PTnPOL bit is used to reverse the polarity of the PWM output waveform.

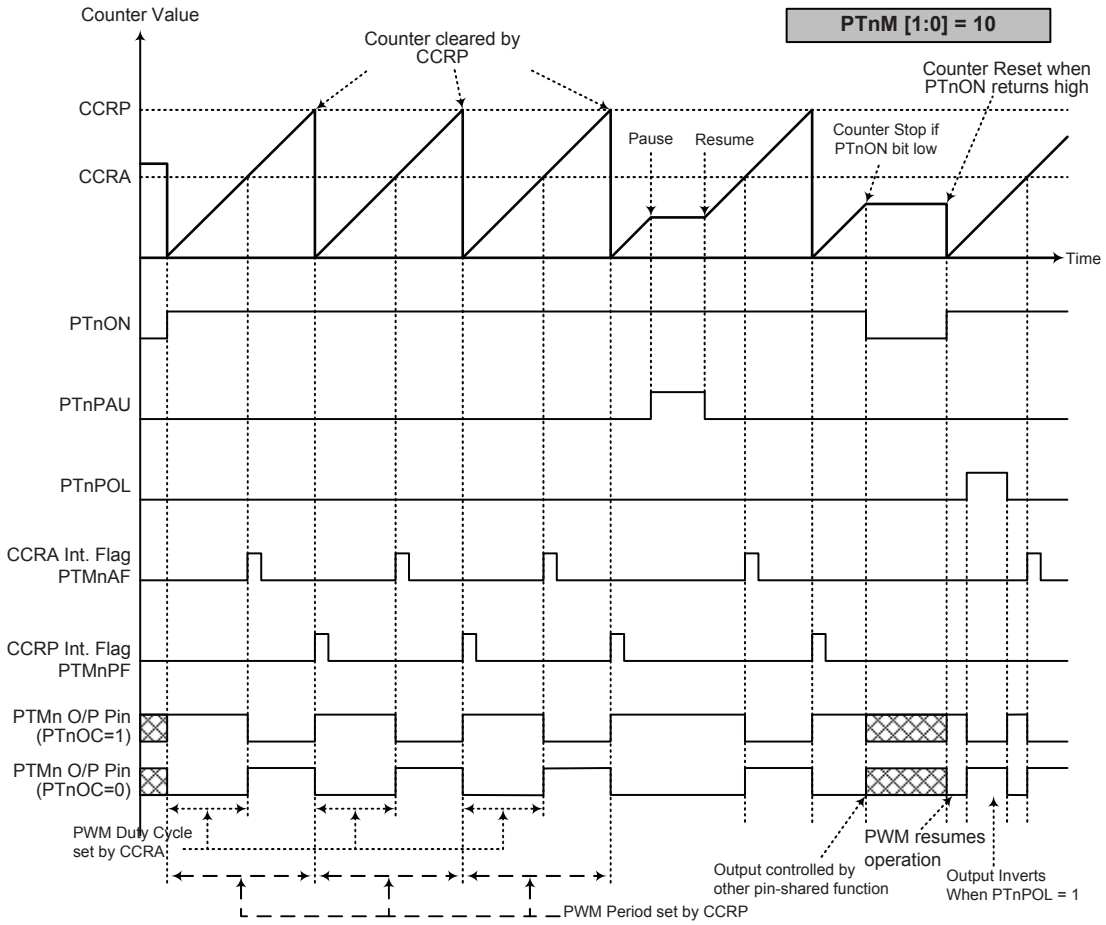
- **10-bit PTMn, PWM Output Mode, Edge-aligned Mode**

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

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

The PTMn PWM output frequency=($f_{SYS}/4$)/512=$f_{SYS}/2048$=5.8594kHz, duty=128/512=25%.

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



PWM Output Mode (n=0, 1 or 2)

Note: 1. Counter cleared by CCRP

2. A counter clear sets the PWM Period

3. The internal PWM function continues running even when PTnIO[1:0]=00 or 01

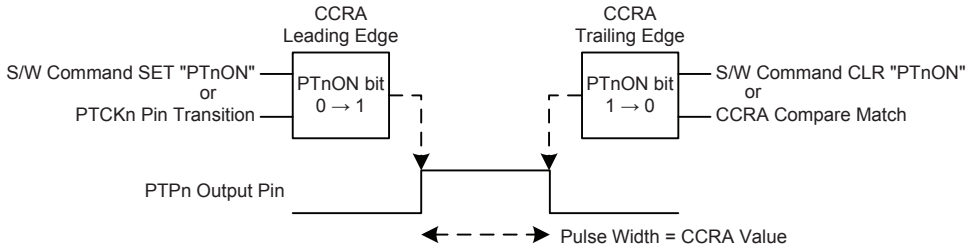
4. The PTnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

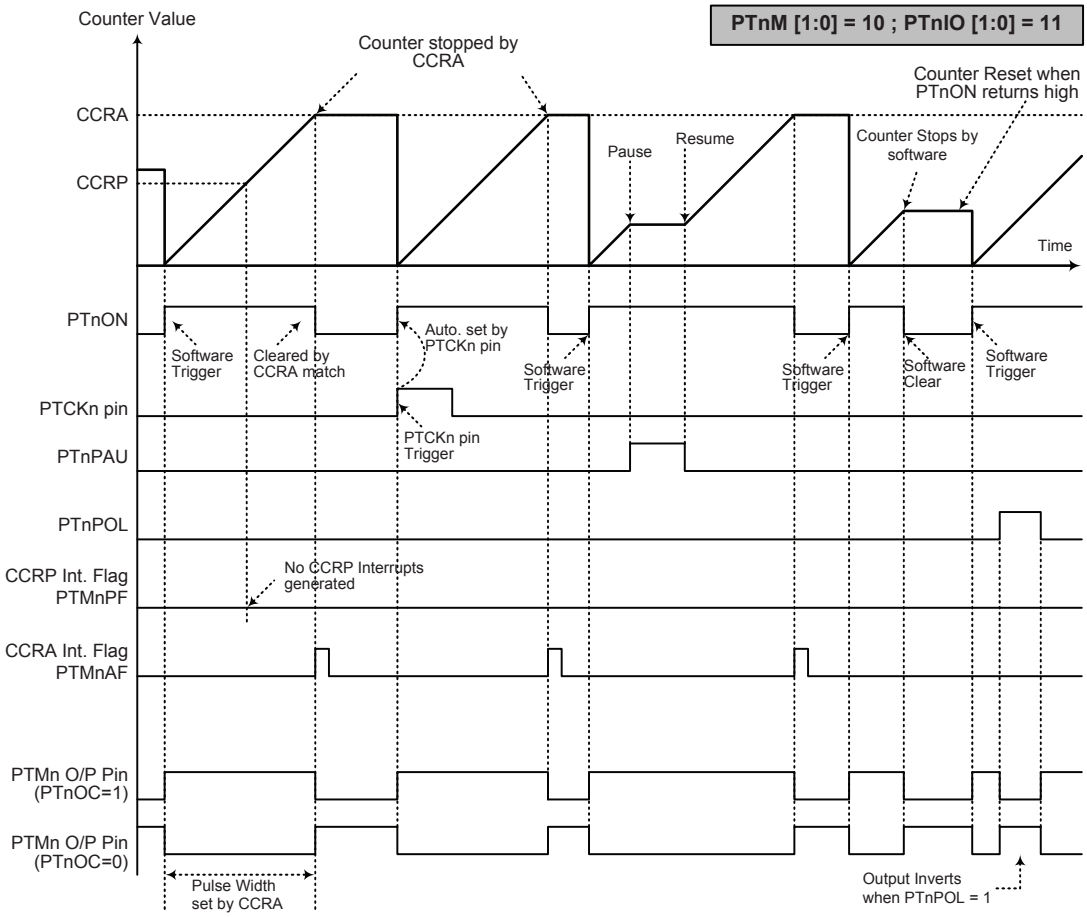
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to "10" respectively and also the PTnIO1 and PTnIO0 bits should be set to "11" respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

The trigger for the pulse output leading edge is a low to high transition of the PTnON bit, which can be implemented using the application program. However in the Single Pulse Mode, the PTnON bit can also be made to automatically change from low to high using the external PTCKn pin, which will in turn initiate the Single Pulse output. When the PTnON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The PTnON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the PTnON bit is cleared to "0", which can be implemented using the application program or when a compare match occurs from Comparator A.

However a compare match from Comparator A will also automatically clear the PTnON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate a PTMn interrupt. The counter can only be reset back to "0" when the PTnON bit changes from low to high when the counter restarts. In the Single Pulse Output Mode CCRP is not used. The PTnCCLR bit is not used in this Mode.



Single Pulse Generation (n=0, 1 or 2)



Single Pulse Output Mode (n=0, 1 or 2)

Note: 1. Counter stopped by CCRA

2. CCRP is not used

3. The pulse is triggered by the PTCKn pin or by setting the PTnON bit high

4. A PTCKn pin active edge will automatically set the PTnON bit high

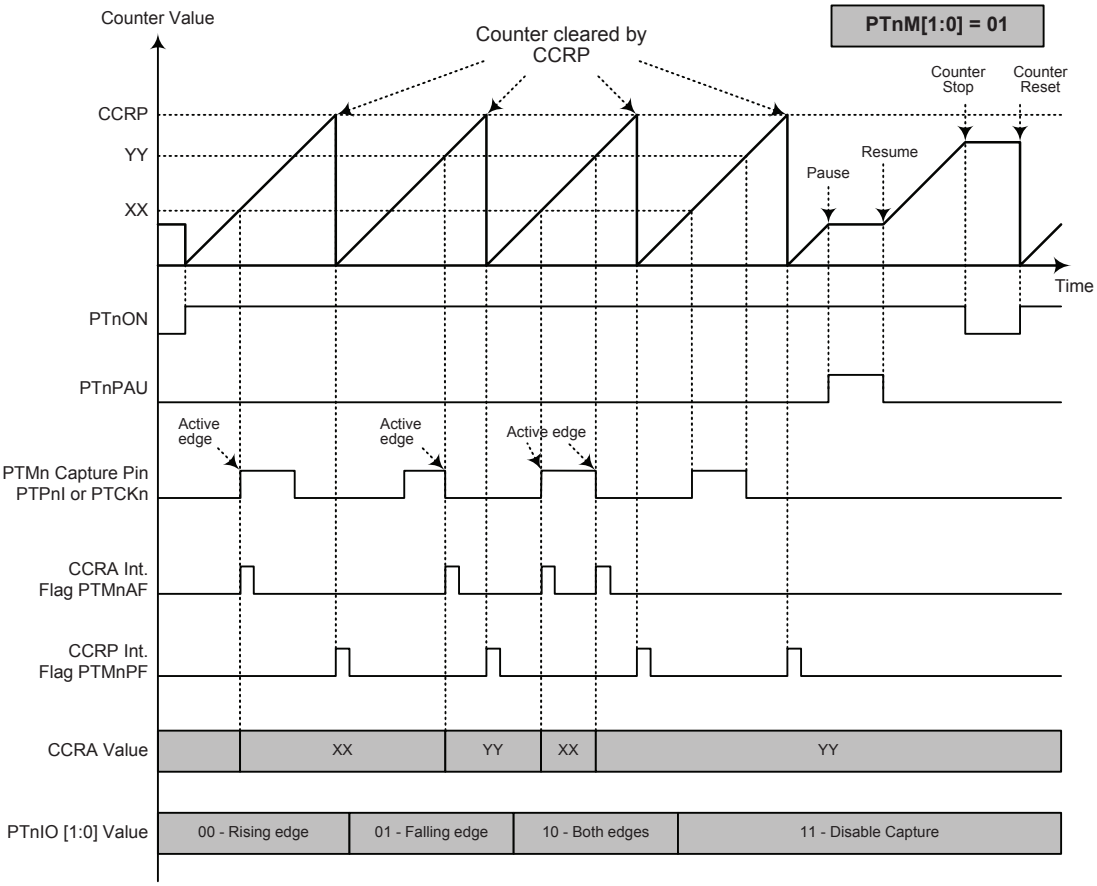
5. In the Single Pulse Output Mode, PTnIO[1:0] must be set to “11” and cannot be changed.

Capture Input Mode

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

When the required edge transition appears on the PTPnI or PTCKn pin the present value in the counter will be latched into the CCRA registers and a PTMn interrupt generated. Irrespective of what events occur on the PTPnI or PTCKn pin, the counter will continue to free run until the PTnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to "0"; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTMn interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTnIO1 and PTnIO0 bits can select the active trigger edge on the PTPnI or PTCKn pin to be a rising edge, falling edge or both edge types. If the PTnIO1 and PTnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPnI or PTCKn pin, however it must be noted that the counter will continue to run.

As the PTPnI or PTCKn pin is pin shared with other functions, care must be taken if the PTMn is in the Capture Input Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The PTnCCLR, PTnOC and PTnPOL bits are not used in this Mode.



Capture Input Mode (n=0, 1 or 2)

Note: 1. PTnM[1:0]=01 and active edge set by the PTnIO[1:0] bits

2. A PTMn Capture input pin active edge transfers the counter value to CCRA

3. PTnCCLR bit not used

4. No output function – PTnOC and PTnPOL bits are not used

5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to "0".

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.

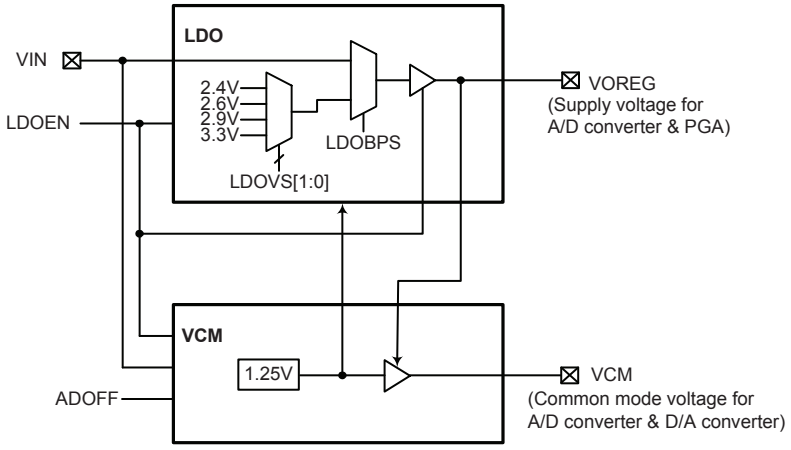
A/D Overview

Each device contains a high accuracy multi-channel 24-bit Delta Sigma Analog-to-Digital Converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 24-bit digital value.

In addition, the PGA gain control, A/D converter gain control and A/D converter reference gain control determine the amplification gain for A/D converter input signal. The designer can select the best gain combination for the desired amplification applied to the input signal. The following block diagram illustrates the A/D converter basic operational function. The A/D converter input channel can be arranged as four single-ended A/D input channels or two differential input channels. The input signal can be amplified by PGA before entering the 24-bit Delta Sigma A/D converter. The Delta Sigma A/D converter modulator will output one bit converted data to SINC filter which can transform the converted one-bit data to 24 bits and store them into the specific data registers. Additionally, the devices also provide a temperature sensor to compensate the A/D converter deviation caused by the temperature. With high accuracy and performance, the devices are very suitable for the Weight Scale related products.

Internal Power Supply

Each device contains an LDO and VCM for the regulated power supply. The accompanying block diagram illustrates the basic functional operation. The internal LDO can provide the fixed voltage for PGA, A/D converter or the external components; as well the V_{CM} can be used as the reference voltage for A/D converter module. There are four LDO voltage levels, 2.4V, 2.6V, 2.9V or 3.3V, decided by LDOVS1~LDOVS0 bits in the PWRC register, as well the VCM voltage is fixed at 1.25V. The LDO and VCM functions can be controlled by the LDOEN bit and can be powered off to reduce the power consumption. If the VCM is disabled, the VCM output pin is floating.



Internal Power Supply Block Diagram

Register bits		Output Voltage		
ADOFF	LDOEN	Bandgap	VOREG	VCM
1	0	Off	Disable	Disable
1	1	On	Enable	Enable
0	0	On	Disable	Enable
0	1	On	Enable	Enable

Power Control Table

PWRC Register

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

Bit 7 **LDOEN**: LDO function control bit
0: Disable
1: Enable
If the LDO is disabled, there will be no power consumption and LDO output will be in a low level by a weakly pull-low resistor.

Bit 6~3 Unimplemented, read as "0"

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

Bit 1~0 **LDOVS1~LDOVS0**: LDO output voltage selection
00: 2.4V
01: 2.6V
10: 2.9V
11: 3.3V

A/D Data Rate Definition

The Delta Sigma A/D converter data rate can be calculated using the following equation:

$$\text{Data Rate} = \frac{f_{ADCK}}{CHOP \times OSR} = \frac{f_{MCLK}/N}{CHOP \times OSR} = \frac{f_{MCLK}}{N \times CHOP \times OSR}$$

f_{ADCK}: A/D clock input, derived from f_{MCLK}/N

f_{MCLK}: A/D clock source, derived from f_{SYS} or $f_{SYS}/2/(ADCK+1)$ using the ADCK bit field.

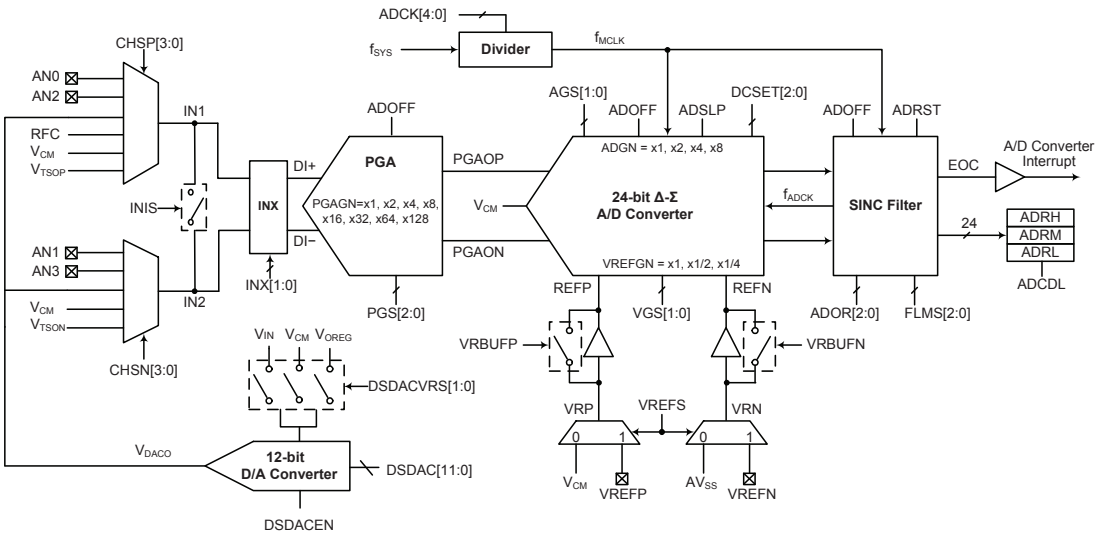
N: a constant divided factor that can be equal to 30 or 12 determined by the FLMS bit field.

CHOP: Sampling data amount doubling function control and can be equal to 2 or 1 determined by the FLMS bit field.

OSR: Oversampling rate determined by the ADOR field.

For example，if a data rate of 8Hz is desired，an f_{MCLK} clock source with a frequency of 4MHz A/D converter can be selected. Then set the FLMS field to "000" to obtain an "N" equal to 30 and "CHOP" equal to 2. Finally, set the ADOR field to "001" to select an oversampling rate equal to 8192. Therefore, the Data Rate = 4MHz / (30 × 2 × 8192) = 8Hz.

Note that the A/D converter has a notch rejection function for an AC power supply with a frequency of 50Hz or 60Hz when the data rate is equal to 10Hz.



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled by using 13 registers. Three read only registers exist to store the A/D converter data 24-bit value. A control register named as PWRC is used to control the required bias and supply voltages for PGA and A/D converter and is described in the "Internal Power Supply" section. Three registers are D/A converter control registers. The remaining 6 registers are control registers which set up the gain selections and control functions of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PGAC0	—	VGS1	VGS0	AGS1	AGS0	PGS2	PGS1	PGS0
PGAC1	—	INIS	INX1	INX0	DCSET2	DCSET1	DCSET0	—
PGACS	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
ADRL	D7	D6	D5	D4	D3	D2	D1	D0
ADRM	D15	D14	D13	D12	D11	D10	D9	D8
ADRH	D23	D22	D21	D20	D19	D18	D17	D16
ADCR0	ADRST	ADSLP	ADOFF	ADOR2	ADOR1	ADOR0	—	VREFS
ADCR1	FLMS2	FLMS1	FLMS0	VRBUFN	VRBUFP	ADCDL	EOC	—
ADCS	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
DSDAH	D11	D10	D9	D8	D7	D6	D5	D4
DSDAL	—	—	—	—	D3	D2	D1	D0
DSDACC	DSDACEN	DSDACVRS1	DSDACVRS0	—	—	—	—	—

A/D Converter Register List

Programmable Gain Amplifier – PGA

There are three registers related to the programmable gain control, PGAC0, PGAC1 and PGACS. The PGAC0 resister is used to select the PGA gain, A/D converter gain and the A/D converter reference gain. As well, the PGAC1 register is used to define the input connection and differential input offset voltage adjustment control. In addition, The PGACS register is used to select the input ends for the PGA. Therefore, the input channels have to be determined by the CHSP[3:0] and CHSN[3:0] bits to determine which analog channel input pins, temperature detector inputs or internal power supply are actually connected to the internal differential A/D converter.

PGAC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	VGS1	VGS0	AGS1	AGS0	PGS2	PGS1	PGS0
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 6~5 **VGS1~VGS0**: REFP/REFN Differential Reference Voltage Gain Selection
00: VREFGN=1
01: VREFGN=1/2
10: VREFGN=1/4
11: Reserved

Bit 4~3 **AGS1~AGS0**: A/D Converter PGAOP/PGAON Differential Input Signal Gain Selection
00: ADGN=1
01: ADGN=2
10: ADGN=4
11: ADGN=8

Bit 2~0 **PGS2~PGS0**: PGA DI+/DI- Differential Channel Input Gain Selection
000: PGAGN=1
001: PGAGN=2
010: PGAGN=4
011: PGAGN=8
100: PGAGN=16
101: PGAGN=32
110: PGAGN=64
111: PGAGN=128

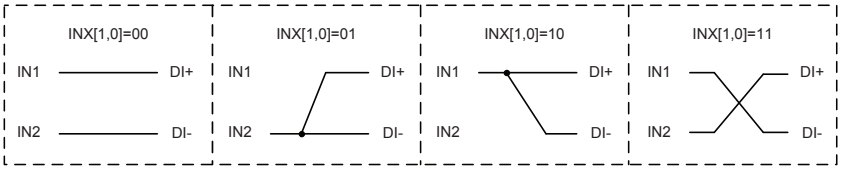
PGAC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	INIS	INX1	INX0	DCSET2	DCSET1	DCSET0	—
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	—	0	0	0	0	0	0	—

Bit 7 Unimplemented, read as "0"

Bit 6 **INIS**: The Selected Input Ends, IN1 and IN2 Internal Connection Control Bit
0: Not connected
1: Connected

Bit 5~4 **INX1, INX0**: The Selected Input Ends, IN1/IN2 and the PGA Differential Input Ends, DI+/DI- Connection Control Bits



Bit 3~1 **DCSET2~DCSET0**: Differential Input Signal PGAOP/PGAON Offset Selection
000: DCSET = +0V
001: DCSET = +0.25×ΔVR_I
010: DCSET = +0.5×ΔVR_I
011: DCSET = +0.75×ΔVR_I
100: DCSET = +0V
101: DCSET = -0.25×ΔVR_I
110: DCSET = -0.5×ΔVR_I
111: DCSET = -0.75×ΔVR_I
The voltage, ΔVR_I, is the differential reference voltage which is amplified by specific gain selection based on the selected inputs.

Bit 0 Unimplemented, read as "0"

PGACS Register

Bit	7	6	5	4	3	2	1	0
Name	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~4 **CHSN3~CHSN0**: Negative Input End IN2 Selection
0000: AN1
0001: AN3
0010: Reserved
0011: Reserved
0100: Reserved
0101: V_{DACO}
0110: V_{CM}
0111: Temperature sensor output – V_{TSON}
1xxx: Reserved

These bits are used to select the negative input, IN2. If the IN2 input is selected as a single end input, the V_{CM} voltage must be selected as the positive input on IN1 for single end input applications. It is recommended that when the V_{TSON} signal is selected as the negative input, the V_{TSOP} signal should be selected as the positive input for proper operations.

Bit 3~0 **CHSP3~CHSP0**: Positive Input End IN1Selection
0000: AN0
0001: AN2
0010: Reserved
0011: Reserved
0100: Reserved
0101: V_{DACO}
0110: V_{CM}
0111: Temperature sensor output – V_{TSOP}
1xxx: RFC

These bits are used to select the positive input, IN1. If the IN1 input is selected as a single end input, the V_{CM} voltage must be selected as the negative input on IN2 for single end input applications. It is recommended that when the V_{TSOP} signal is selected as the positive input, the V_{TSON} signal should be selected as the negative input for proper operations

D/A Converter Registers – DSDAH, DSDAL, DSDACC

There are three registers related to the D/A converter output control.

- **DSDAH 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 **D11~D4**: D/A Converter Output Control Code, Only for 12 bits D/A Converter

- **DSDAL 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 Unimplemented, read as "0"

Bit 3~0 **D3~D0**: D/A Converter Output Control Code

Note: writing this register only writes to shadow buffer, and until write DSDAH register will also copy the shadow buffer data to DSDAL register.

- **DSDACC Register**

Bit	7	6	5	4	3	2	1	0
Name	DSDACEN	DSDACVRS1	DSDACVRS0	—	—	—	—	—
R/W	R/W	R/W	R/W	—	—	—	—	—
POR	0	0	0	—	—	—	—	—

Bit 7 **DSDACEN**: D/A Converter Enable or Disable Control Bit
0: Disable
1: Enable

Bit 6~5 **DSDACVRS1~DSDACVRS0**: D/A Converter Reference Voltage Selection
00: D/A converter reference voltage comes from V_{OREG}
01: D/A converter reference voltage comes from V_{IN}
1x: D/A converter reference voltage comes from V_{CM}

Bit 4~0 Unimplemented, read as "0"

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

Each device contains an internal 24-bit Delta Sigma A/D Converter, it requires three data registers to store the converted value. These are a high byte register, known as ADRH, a middle byte register, known as ADRM, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. D0~D23 are the A/D conversion result data bits.

- **ADRL Register**

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

"x" unknown

Bit 7~0 **D7~D0**: A/D Conversion Data Register bit 7~bit 0

- **ADRM Register**

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

"x" unknown

Bit 7~0 **D15~D8**: A/D Conversion Data Register bit 15~bit 8

- **ADRH Register**

Bit	7	6	5	4	3	2	1	0
Name	D23	D22	D21	D20	D19	D18	D17	D16
R/W	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x

"x" unknown

Bit 7~0 **D23~D16**: A/D Conversion Data Register bit 23~bit 16

A/D Converter Control Registers − ADCR0, ADCR1, ADCS

To control the function and operation of the A/D converter, three control registers known as ADCR0, ADCR1 and ADCS are provided. These 8-bit registers define functions such as the selection of which reference source is used for the internal A/D converter, the A/D converter clock source, the A/D converter output data rate as well as controlling the power-up function and monitoring the A/D converter end of conversion status.

- **ADCR0 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADRST	ADSLP	ADOFF	ADOR2	ADOR1	ADOR0	—	VREFS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	R/W
POR	0	0	1	0	0	0	—	0

Bit 7 **ADRST**: A/D Converter Software Reset Control bit.
0: Disable
1: Enable
This bit is used to reset the A/D converter internal digital SINC filter. This bit is set low for A/D normal operations. However, if set high, the internal digital SINC filter will be reset and the current A/D converted data will be aborted. A new A/D data conversion process will not be initiated until this bit is set low again.

Bit 6 **ADSLP**: A/D Converter Sleep Mode Control bit
0: Normal mode
1: Sleep mode
This bit is used to determine whether the A/D converter enters the sleep mode or not when the A/D converter is powered on by setting the ADOFF bit low. When the A/D converter is powered on and the ADSLP bit is low, the A/D converter will operate normally. However, the A/D converter will enter the sleep mode if the ADSLP bit is set high as the A/D converter has been powered on. The whole A/D converter circuit will be switched off except the PGA and internal Bandgap circuit to reduce the power consumption and V_{CM} start-up stable time.

Bit 5 **ADOFF**: A/D Converter Module Power On/Off Control bit
0: Power on
1: Power off
This bit controls the power of the A/D converter module. This bit should be cleared to “0” to enable the A/D converter. If the bit is set high then the A/D converter will be switched off reducing the device power consumption. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.
It is recommended to set the ADOFF bit high before the device enters the IDLE/SLEEP mode for saving power. Setting the ADOFF bit high will power down the A/D converter module regardless of the ADSLP and ADRST bit settings.

Bit 4~2 **ADOR2~ADOR0**: A/D Conversion Oversampling Rate Selection
000: Oversampling rate OSR=16384
001: Oversampling rate OSR=8192
010: Oversampling rate OSR=4096
011: Oversampling rate OSR=2048
100: Oversampling rate OSR=1024
101: Oversampling rate OSR=512
110: Oversampling rate OSR=256
111: Oversampling rate OSR=128

Bit 1 Unimplemented, read as “0”

Bit 0 **VREFS**: A/D Converter Reference Voltage Pair Selection
0: Internal reference voltage pair – V_{CM} & AV_{SS}
1: External reference voltage pair – VREFP & VREFN

- **ADCR1 Register**

Bit	7	6	5	4	3	2	1	0
Name	FLMS2	FLMS1	FLMS0	VRBUFN	VRBUFP	ADCDL	EOC	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	0	0	0	0	0	0	0	—

Bit 7~5 **FLMS2~FLMS0**: A/D Converter Clock Divided Ratio Selection and Sampled Data Doubling Function (CHOP) Enable Control
000: CHOP=2, $f_{ADCK} = f_{MCLK} / 30$
010: CHOP=2, $f_{ADCK} = f_{MCLK} / 12$
Others: Reserved
When the CHOP bit is equal to 2, it means that the sampled data amount will be doubled for the normal conversion mode.

Bit 4 **VRBUFN**: A/D Converter Negative Reference Voltage Input (VRN) Buffer Control
0: Disable input buffer and enable bypass function
1: Enable input buffer and disable bypass function

Bit 3 **VRBUFP**: A/D Converter Positive Reference Voltage Input (VRP) Buffer Control
0: Disable input buffer and enable bypass function
1: Enable input buffer and disable bypass function

Bit 2 **ADCDL**: A/D Converted Data Latch Function Enable Control
0: Disable data latch function
1: Enable data latch function
If the A/D converted data latch function is enabled, the latest converted data value will be latched and not be updated by any subsequent converted results until this function is disabled. Although the converted data is latched into the data registers, the A/D converter circuits remain operational, but will not generate interrupt and EOC will not change. It is recommended that this bit should be set high before reading the converted data in the ADRL, ADRM and ADRH registers. After the converted data has been read out, the bit can then be cleared to low to disable the A/D converter data latch function and allow further conversion values to be stored. In this way, the possibility of obtaining undesired data during A/D converter conversions can be prevented.

Bit 1 **EOC**: End of A/D Conversion Flag
0: A/D conversion in progress
1: A/D conversion ended
This bit must be cleared by software.

Bit 0 Unimplemented, read as “0”

- **ADCS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **ADCK4~ADCK0**: A/D Converter Clock Source f_{MCLK} Divided Ratio Selection
00000~11110: $f_{MCLK}=f_{SYS}/2$ / (ADCK[4:0]+1)
11111: $f_{MCLK}=f_{SYS}$

A/D Operation

The A/D converter provides three operational modes, which are Power down mode, Sleep mode and Reset mode, controlled respectively by the ADOFF, ADSLP and ADRST bits in the ADCR0 register. The following table illustrates the operating mode selection.

LDOEN	ADOFF	ADSLP	ADRST	Operating Mode	Description
0	1	x	x	Power down mode	Bandgap off, LDO off, V_{CM} generator off, PGA off, ADC off, Temperature sensor off, VRN/VRP buffer off, SINC filter off
1	1	x	x	Power down mode	Bandgap on, LDO on, V_{CM} generator off, PGA off, ADC off, Temperature sensor off, VRN/VRP buffer off, SINC filter off
0	0	1	x	Sleep mode (External voltage must be supplied on LDO output pin)	Bandgap on, LDO off, V_{CM} generator off, PGA on, ADC off, Temperature sensor off, VRN/VRP buffer off, SINC filter on
0	0	0	0	Normal mode (External voltage must be supplied on LDO output pin)	Bandgap on, LDO off, V_{CM} generator on/off(1), PGA on, ADC on, Temperature sensor on/off(2), VRN/VRP buffer on/off(3), SINC filter on
0	0	0	1	Reset mode (External voltage must be supplied on LDO output pin)	Bandgap on, LDO off, V_{CM} generator on/off(1), PGA on, ADC on,Temperature sensor on/off(2), VRN/VRP buffer on/off(3), SINC filter reset
1	0	1	x	Sleep mode	Bandgap on, LDO on, V_{CM} generator off, PGA on, ADC off, Temperature sensor off, VRN/VRP buffer off, SINC filter on
1	0	0	0	Normal mode	Bandgap on, LDO on, V_{CM} generator on/off(1), PGA on, ADC on, Temperature sensor on/off(2), VRN/VRP buffer on/off(3), SINC filter on
1	0	0	1	Reset mode	Bandgap on, LDO on, V_{CM} generator on/off(1), PGA on, ADC on, Temperature sensor on/off(2), VRN/VRP buffer on/off(3), SINC filter reset

"x" unknown

Note: 1. The V_{CM} generator can be switched on or off by the bandgap on or off.
2. The Temperature sensor can be switched on or off by configuring the CHSN[3:0] or CHSP[3:0] bits
3. The VRN buffer can be switched on or off by configuring the VRBUFN bit while the VRP buffer can be switched on or off by configuring the VRBUFP bit

A/D Operation Mode Selection

To enable the A/D converter, the first step is to disable the A/D converter power down and sleep mode by clearing the ADOFF and ADSLP bits to make sure the A/D converter is powered up. The ADRST bit in the ADCR0 register is used to start and reset the A/D converter after power on. To set ADRST bit from low to high and then low again, an analog to digital converted data in SINC filter will be initiated. After this setup is complete, the A/D Converter is ready for operation. These three bits are used to control the overall start operation of the internal analog to digital converter.

The EOC bit in the ADCR1 register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to "1" by the Hardware 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 poll the EOC bit in the ADCR1 register to check whether it has been set "1" as an alternative method of detecting the end of an A/D conversion cycle. The A/D converted data will be updated continuously by the new converted data. If the A/D converted data latch function is enabled, the latest converted data will be latched and the following new converted data will be discarded until this data latch function is disabled.

The clock source for the A/D converter should be typically fixed at a value of 4MHz, which originates from the system clock f_{SYS}, and can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the ADCK4~ADCK0 bits in the ADCS register to obtain a 4MHz clock source for the A/D converter.

The differential reference voltage supply to the A/D Converter can be supplied from either the internal power supply, V_{CM} and AV_{SS}, or from an external reference source, VREFP and VREFN. The desired selection is made using the VREFS bit in the ADCR0 register.

Summary of A/D Conversion Steps

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

- Step 1
  Enable the power LDO, VCM for PGA and A/D converter.
- Step 2
  Select the PGA, A/D converter, reference voltage gains by PGAC0 register
- Step 3
  Select the PGA settings for input connection, V_{CM} voltage level and buffer option by PGAC1 register
- Step 4
  Select the required A/D conversion clock source by correctly programming bits ADCK4~ADCK0 in the ADCS register.
- Step 5
  Select output data rate by configuring the ADOR[2:0] bits in the ADCR0 register and FLMS[2:0] bits in the ADCR1 register.
- Step 6
  Select which channel is to be connected to the internal PGA by correctly programming the CHSP3~CHSP0 and CHSN3~CHSN0 bits which are also contained in the PGACS register.
- Step 7
  Release the power down mode and sleep mode by clearing the ADOFF and ADSLP bits in ADCR0 register.

- Step 8
  Reset the A/D by setting the ADRST to high in the ADCR0 register and clearing this bit to "0" to release reset status.
- Step 9
  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, and the A/D converter interrupt bit, ADE, must both be set high to do this.
- Step 10
  To check when the analog to digital conversion process is complete, the EOC bit in the ADCR1 register can be polled. The conversion process is complete when this bit goes high. When this occurs the A/D data registers ADRL, ADRM 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 EOC bit in the ADCR1 register is used, the interrupt enable step above can be omitted.

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D 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.

A/D Converter Transfer Function

Each device contains a 24-bit Delta Sigma A/D converter and its full-scale converted digitised value is from 8388607 to -8388608 in decimal value. The converted data format is formed by a two's complement binary value. The MSB of the converted data is the signed bit. Since the full-scale analog input value is equal to the amplified value of the V_{CM} or differential reference input voltage, ΔVR_I, selected by the VREFS bit in ADCR0 register, this gives a single bit analog input value of ΔVR_I divided by 8388608.

$$1\ LSB = \Delta VR\_I / 8388608$$

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

$$\Delta SI\_I = (PGAGN \times ADGN \times \Delta DI\pm) + DCSET$$

$$\Delta VR\_I = VREGN \times \Delta VR\pm$$

$$ADC\_Conversion\_Data = (\Delta SI\_I / \Delta VR\_I) \times K$$

Where K is equal to 2^{23}

Note: 1. The PGAGN, ADGN, VREGN values are decided by the PGS, AGS, VGS control bits.
2. ΔSI_I: Differential Input Signal after amplification and offset adjustment.
3. PGAGN: Programmable Gain Amplifier gain
4. ADGN: A/D Converter gain
5. VREGN: Reference voltage gain
6. ΔDI±: Differential input signal derived from external channels or internal signals
7. DCSET: Offset voltage
8. ΔVR±: Differential Reference voltage
9. ΔVR_I: Differential Reference input voltage after amplification

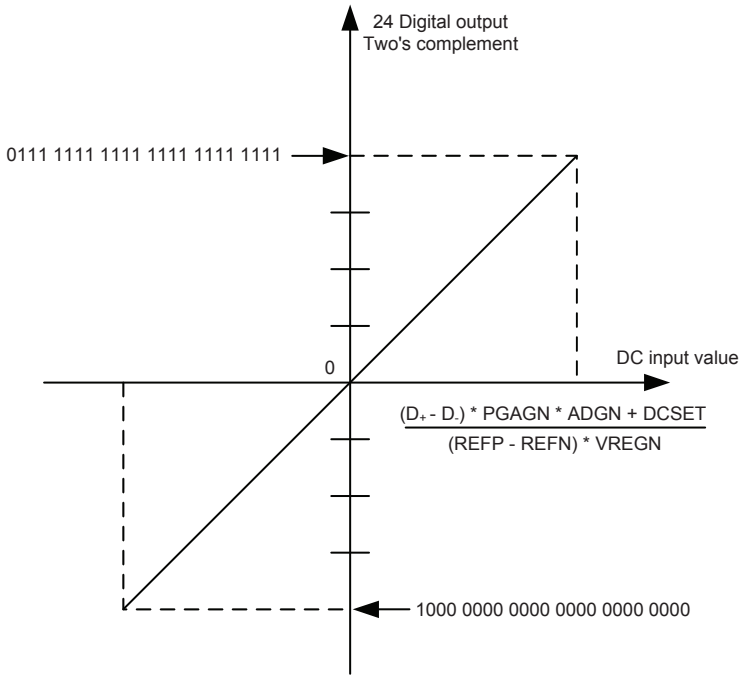
Due to the digital system design of the Delta Sigma A/D Converter, the maximum number of the A/D converted value is 8388607 and the minimum value is -8388608, therefore, we can have the middle number 0. The ADC_Conversion_Data equation illustrates this range of converted data variation.

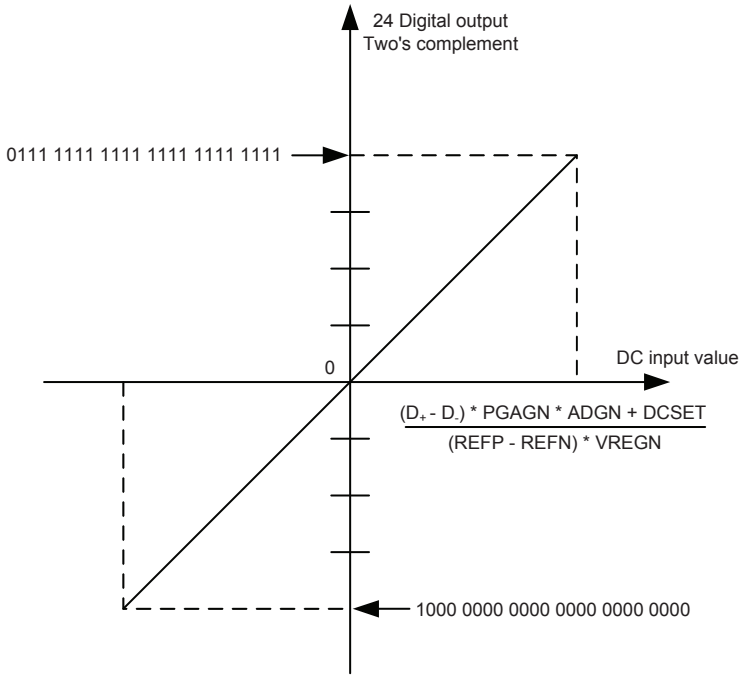
A/D Conversion Data (2's Compliment, Hexadecimal)	Decimal Value
0x7FFFFF	8388607
0x800000	-8388608

A/D Conversion Data Range

The above A/D converter conversion data table illustrates the range of A/D conversion data.

The following diagram shows the relationship between the DC input value and the A/D converted data which is presented by the Two's Complement.





A/D Converted Data

The A/D converted data is related to the input voltage and the PGA selections. The format of the A/D converter output is a two's complement binary code. The length of this output code is 24 bits and the MSB is a signed bit. When the MSB is "0", which represents the input is "positive", on the other hand, as the MSB is "1", it represents the input is "negative". The maximum value is 8388607 and the minimum value is -8388608. If the input signal is over the maximum value, the converted data is limited by the 8388607, and if the input signal is less than the minimum value, the converted data is limited by -8388608.

A/D Converted Data to Voltage

The designer can recover the converted data by the following equations:

If MSB=0 (Positive Converted data): Input Voltage=

$$\frac{(\text{Converted\_data}) \times \text{LSB} - \text{DCSET}}{\text{PGA} \times \text{ADGN}}$$

If MSB=1(Negative Converted data): Input voltage=

$$\frac{(\text{Two's\_complement\_of\_Converted\_data}) \times \text{LSB} - \text{DCSET}}{\text{PGA} \times \text{ADGN}}$$

Note: Two's complement=One's complement +1

A/D Programming Example

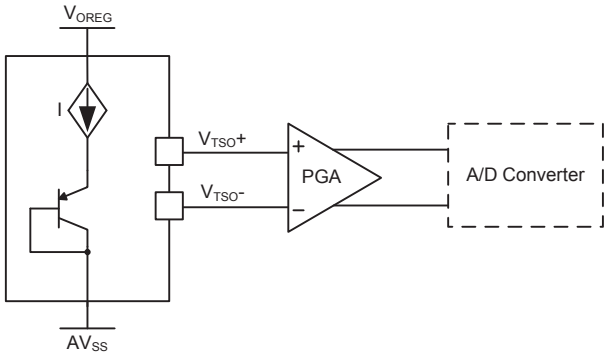
- **Example: Using an EOC polling method to detect the end of conversion**

```
#include bh66f2650.inc
data .section   'data'
     adc_result_data_l db ?
     adc_result_data_m db ?
     adc_result_data_h db ?
code .section   'code'
start:
     clr    ADE                 ; Disable A/D converter interrupt
     mov    a, 083H             ; Power control for PGA, A/D converter
     mov    PWRC, a             ; PWRC=10000011, LDO enable, VCM enable, LDO Bypass
                                ; disable, LDO output voltage: 3.3V
     mov    a, 000H
     mov    PGAC0, a            ; PGA gain=1, A/D converter gain=1, VREF gain=1
     mov    a, 000H
     mov    PGAC1, a            ; VCM=1.25V, INIS, INX, DCSET in default value
     set    VRBUFP              ; enable buffer for VREF+
     set    VRBUFN              ; enable buffer for VREF-
     set    VREFS               ; for using external reference
     clr    ADOR2               ; for 10Hz output data rate, ADOR[2:0]=001, FLMS[2:0]=000
     clr    ADOR1
     set    ADOR0
     clr    FLMS2
     clr    FLMS1
     clr    FLMS0

     clr    ADOFF               ; A/D converter exit power down mode.
     set    ADRST               ; A/D converter in reset mode
     clr    ADRST               ; A/D converter in convertsion (continuos mode)
     clr    EOC                 ; Clear "EOC" flag
loop:
     snz    EOC                 ; Polling "EOC" flag
     jmp    loop                ; Wait for read data
     clr    adc_result_data_h
     clr    adc_result_data_m
     clr    adc_result_data_l
     mov    a, ADRL
     mov    adc_result_data_l, a  ; Get Low byte A/D converter value
     mov    a, ADRM
     mov    adc_result_data_m, a  ; Get Middle byte A/D converter value
     mov    a, ADRH
     mov    adc_result_data_h, a  ; Get High byte A/D converter value
get_adc_value_ok:
     clr    EOC                 ; Clearing read flag
     jmp    loop                ; for next data read
end
```

Temperature Sensor

The devices provide an internal temperature sensor to compensate the device due to temperature effects. By selecting the PGA input channels as V_{TSOP} and V_{TSON}, the A/D Converter can obtain temperature information allowing compenstiaon to be made to the A/D converted data.



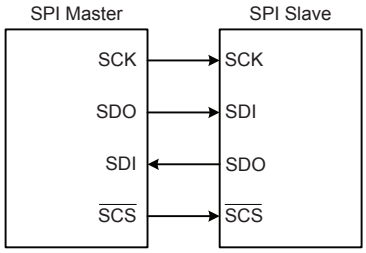
Serial Interface Module – SIM

Each device contains a Serial Interface Module, which includes both the four line SPI interface and the two line I^2C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I^2C based hardware such as sensors, Flash memory, etc. As both interface types share the same pins and registers, the choice of whether the SPI or I^2C type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O pins are selected using pull-high control registers when the SIM function is enabled and the corresponding pins are used as SIM input pins.

SPI Interface

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

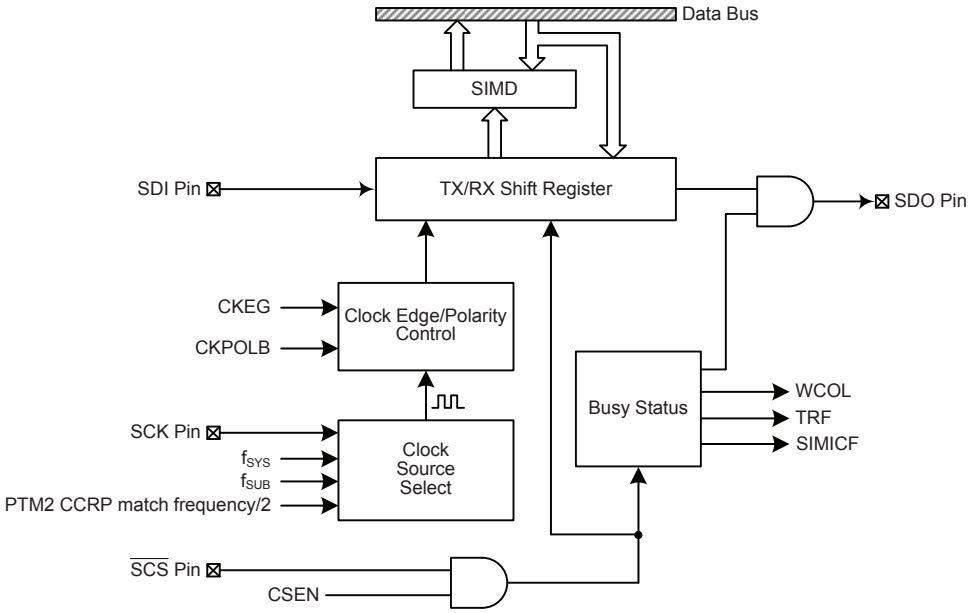
The communication is full duplex and operates as a slave/master type, where the devices can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, but these devices are provided only one $\overline{SCS}$ pin. If the master needs to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



SPI Master/Slave Connection

SPI Interface Operation

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



SPI Block Diagram

The SPI function in these devices offer the following features:

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

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

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two registers SIMC0 and SIMC2. Note that the SIMC1 register is only used by the I²C interface.

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

SIM Registers List

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C functions. Before the device writes data to the SPI bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the SPI bus, the device can read it from the SIMD register. Any transmission or reception of data from the SPI bus must be made via the SIMD register.

- **SIMD Register**

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

"x" unknown

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I²C function. The SIMC1 register is not used by the SPI function, only by the I²C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Although not connected with the SPI function, the SIMC0 register is also used to control the Peripheral Clock Prescaler. Register SIMC2 is used for other control functions such as LSB/MSB selection, write collision flag etc.

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is PTM2 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode
These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from the PTM2. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB[1:0]**: I²C Debounce Time Selection

The SIMDEB[1:0] bits are of no used in SPI mode of SIM, please ignore these selection bits when operate in SPI mode.

Bit 1 **SIMEN**: SIM Control

0: Disable
1: Enable

The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to "0" to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will be in a floating condition and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I²C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SIM SPI Incomplete Flag

0: SIM SPI incompleted is not occurred
1: SIM SPI incompleted is occurred

The SIMICF bit is determined by $\overline{SCS}$ pin. When $\overline{SCS}$ pin is set high, it will clear the SPI counter. Meanwhile, the interrupt is occurred and the incomplete flag, SIMICF, is set high.

- **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 bit

This bit can be read or written by user software program.

Bit 5 **CKPOLB**: Determines the Base Condition of the Clock Line

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**: Determines SPI SCK Active Clock Edge Type

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
1: MSB
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.

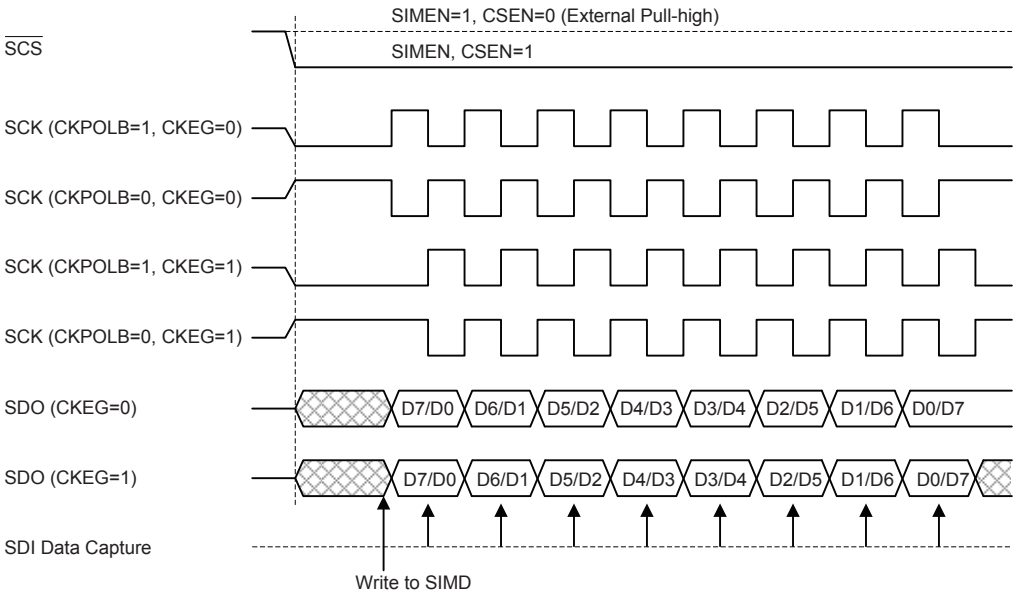
Bit 1 **WCOL**: SPI Write Collision Flag
0: No collision
1: Collision
The WCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program.

Bit 0 **TRF**: SPI Transmit/Receive Complete Flag
0: Data is being transferred
1: SPI data transmission is completed
The TRF bit is the Transmit/Receive Complete flag and is set high automatically when an SPI data transmission is completed, but must cleared to "0" by the application program. It can be used to generate an interrupt.

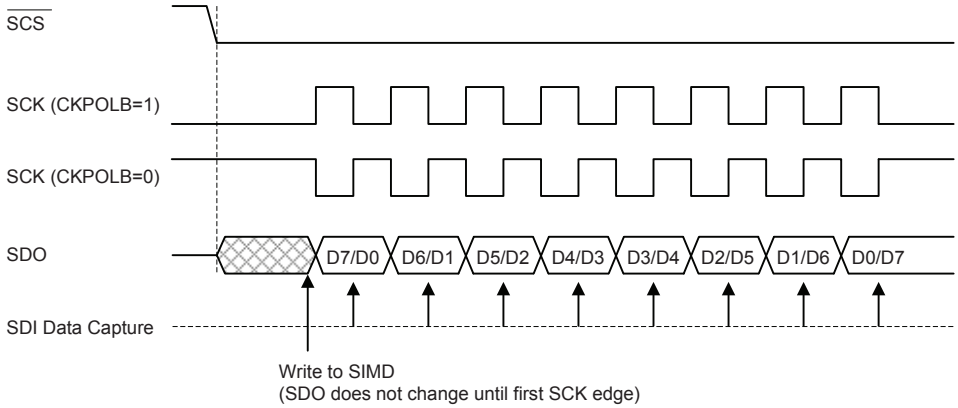
SPI Communication

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

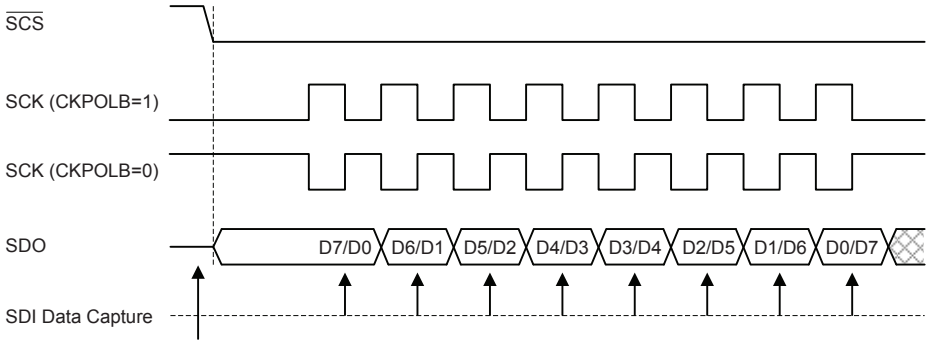
The SPI will continue to function in special IDLE Modes if the clock source used by the SPI interface is still active.



SPI Master Mode Timing

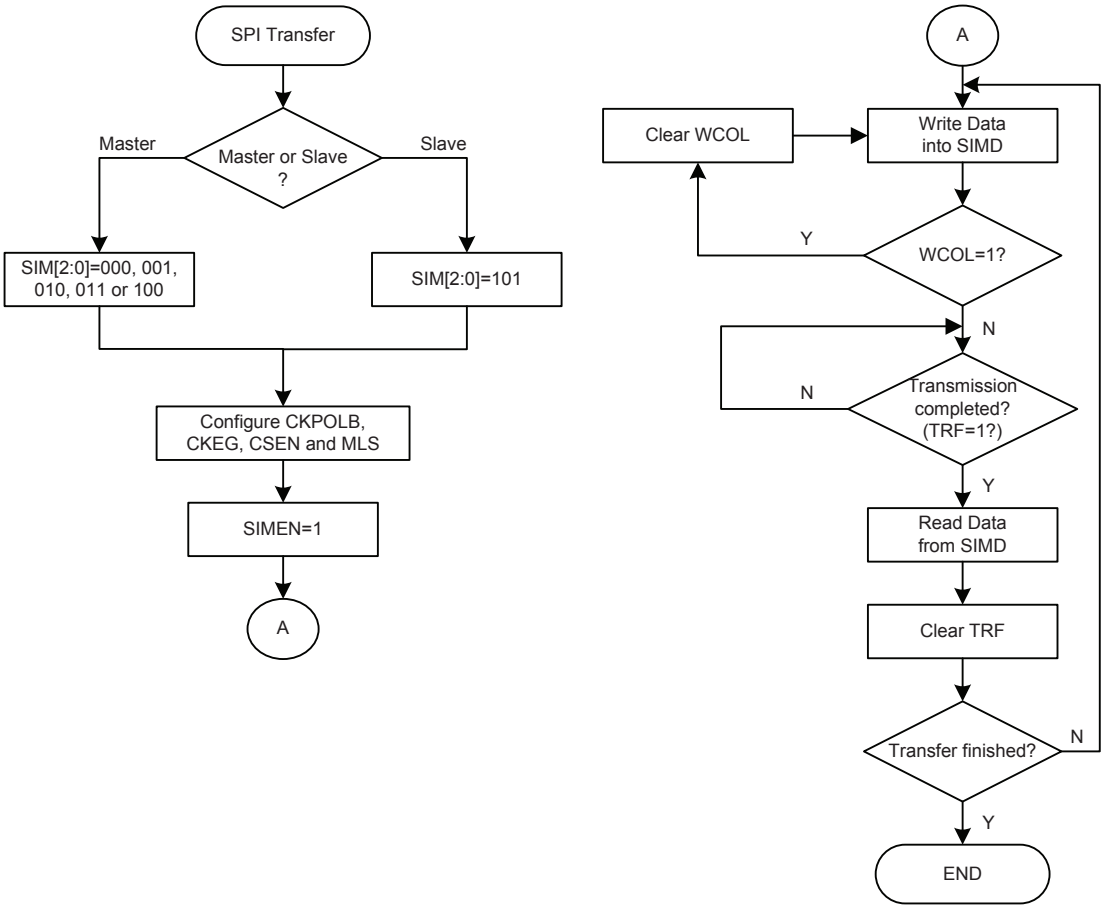


SPI Slave Mode Timing – CKEG=0



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

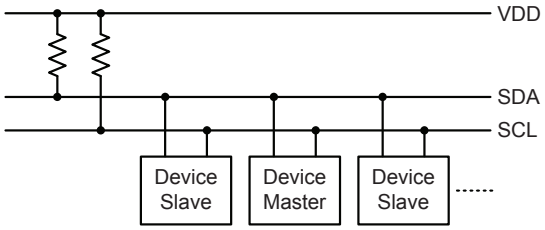
SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flowchart

I²C Interface

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

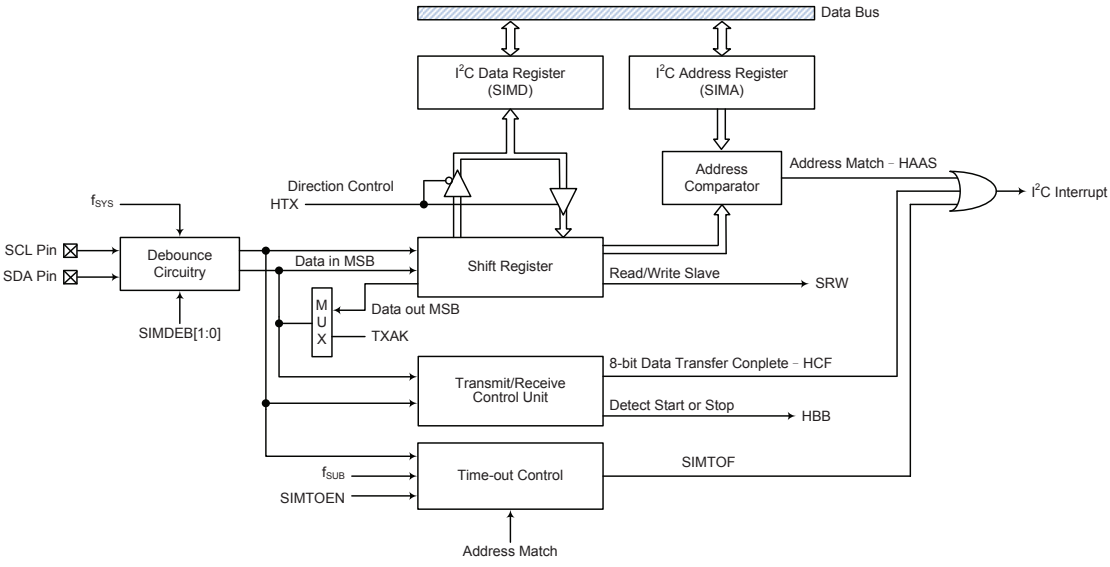


I²C Master/Slave Bus Connection

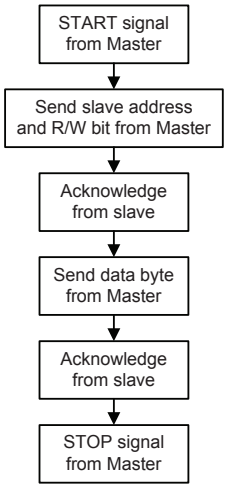
I²C Interface Operation

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

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



I²C Block Diagram



I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMTOC, one address register, SIMA and one data register, SIMD. The SIMD register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I²C bus. Note that the SIMA register also has the name SIMC2 which is used by the SPI function. Bit SIMEN and bits SIM2~SIM0 in register SIMC0 are used by the I²C interface. The SIMTOC register is used for I²C time-out control.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	A6	A5	A4	A3	A2	A1	A0	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C Register List

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is PTM2 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode

These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from the PTM2. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
1x: 4 system clock debounce

Bit 1 **SIMEN**: SIM Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to "0" to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will be in a floating condition and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I²C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SIM SPI Incomplete Flag
SIMICF is of no used in I²C mode of SIM, please ignore this flag when operate in I²C mode.

- **SIMC1 Register**

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

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

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

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

Bit 4 **HTX**: Select I²C Slave Device is Transmitter or Receiver
0: Slave device is the receiver
1: Slave device is the transmitter

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

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

Bit 1 **IAMWU**: I²C Address Match Wake Up Function Control
0: Disable
1: Enable
This bit should be set to "1" to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake up, then this bit must be cleared by the application program after wake-up to ensure correct device operation.

Bit 0 **RXAK**: I²C Bus Receive Acknowledge Flag
0: Slave receives acknowledge flag
1: Slave does not receive acknowledge flag
The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 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 receive wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C bus.

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C functions. Before the device write data to the I²C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I²C bus, the device can read it from the SIMD register. Any transmission or reception of data from the I²C bus must be made via the SIMD register.

- **SIMD Register**

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

"x" unknown

- **SIMA Register**

Bit	7	6	5	4	3	2	1	0
Name	A6	A5	A4	A3	A2	A1	A0	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 **A6~A0**: I²C Slave address

A6~A0 is the I²C slave address bit 6~bit 0.

The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bit 7~Bit 1 of the SIMA register define the device slave address. Bit 0 is not defined.

When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

Bit 0 **D0**: Undefined bit

This bit can be read or written by user software program.

- **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**: I²C interface Time-out control
0: Disable
1: Enable

Bit 6 **SIMTOF**: I²C interface Time-out flag
0: No occurred
1: Occurred

The SIMTOF flag is set by the time-out circuitry when the time-out event occurs and cleared by software program.

Bit 5~0 **SIMTOS5~SIMTOS0**: I²C interface Time-out period selection

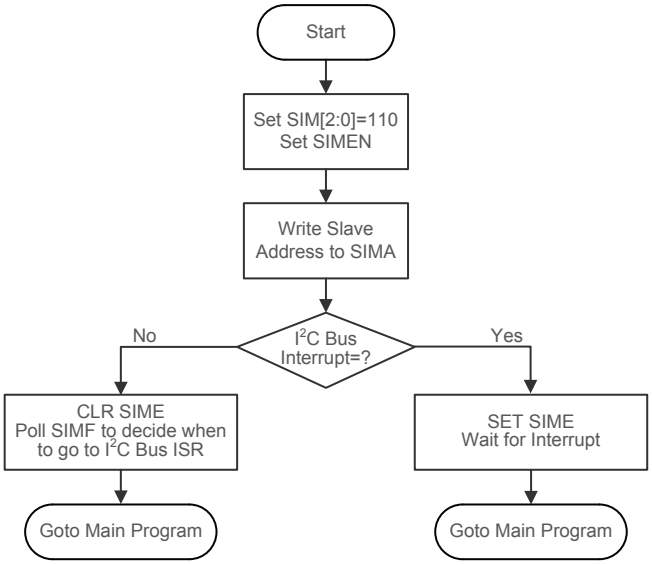
The I²C Time-Out clock source is $f_{SUB}/32$.

The I²C Time-Out time is ([SIMTOS5:SIMTOS0] + 1) × $(32/f_{SUB})$

I²C Bus Communication

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

- Step 1
  Set the SIM2~SIM0 bits to "110" and the SIMEN bits to "1" in the SIMC0 register to enable the I²C bus.
- Step 2
  Write the slave address to the I²C bus address register SIMA.
- Step 3
  Set the interrupt enable bit SIME to enable the SIM interrupt.



I²C Bus Initialisation Flowchart

I²C Bus Start Signal

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

Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the SIMC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

As an I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS bit and SIMTOF bit should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer or from the I²C communication time-out. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

I²C Bus Read/Write Signal

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

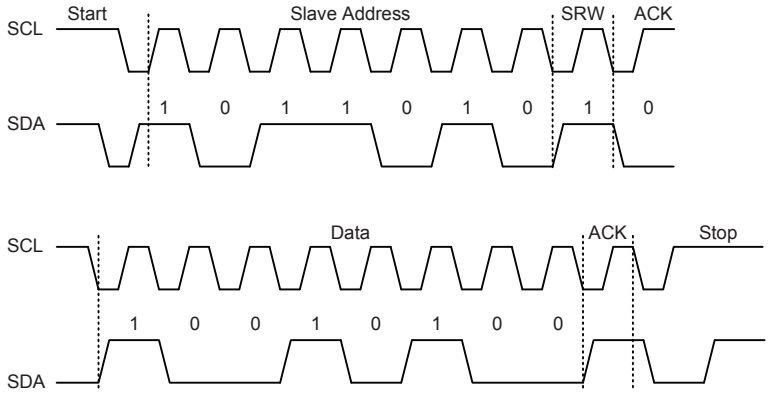
I²C Bus Slave Address Acknowledge Signal

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

I²C Bus Data and Acknowledge Signal

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

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

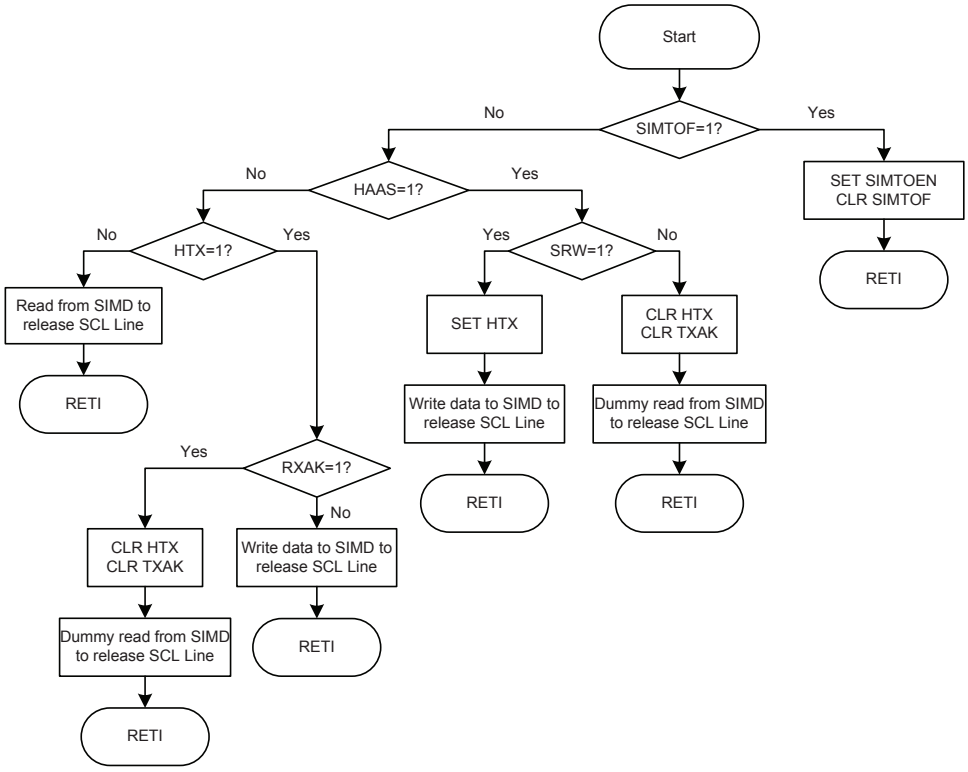


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	·····	S	SA	SR	M	D	A	D	A	·····	P

I²C Communication Timing Diagram

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



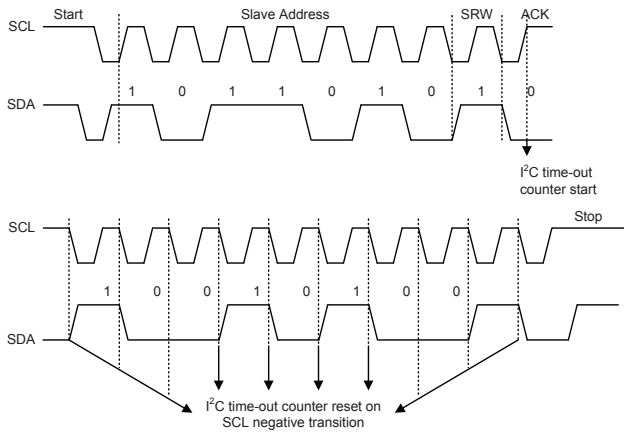
I²C Bus ISR Flowchart

I²C Time-out Function

In order to reduce the I²C lockup problem due to reception of erroneous clock sources, a time-out function is provided. If the clock source connected to the I²C bus is not received for a while, then the I²C circuitry and the SIMC1 register will be reset, the SIMTOF bit in the SIMTOC register will be set high after a certain time-out period. The Time Out function enable/disable and the time-out period are managed by the SIMTOC register.

I²C Time-out Operation

The time-out counter starts to count on an I²C bus "START" & "address match" condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out period specified by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs. There are 64 time-out period selections which can be selected using the SIMTOS0~SIMTOS5 bits in the SIMTOC register.



I²C Time-out Diagram

When an I²C time-out counter overflow occurs, the counter will stop and the SIMTOEN bit will be cleared to "0" and the SIMTOF bit will be set high to indicate that a time-out condition has occurred. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

Registers	After I²C Time-out
SIMD, SIMA, SIMC0	No change
SIMC1	Reset to POR condition

I²C Registers after Time-out

Serial Peripheral Interface – SPIA

Each device contains an independent SPI function. It is important not to confuse this independent SPI function with the additional one contained within the combined SIM function, which is described in another section of this datasheet. This independent SPI function will carry the name SPIA to distinguish it from the other one in the SIM.

The SPIA 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 SPIA interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the programming requirements when communicating with external hardware devices.

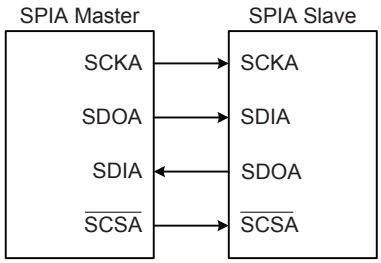
The communication is full duplex and operates as a slave/master type, where the devices can be either master or slave. Although the SPIA interface specification can control multiple slave devices from a single master, however the device is provided with only one $\overline{\text{SCSA}}$ pin. If the master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

SPIA Interface Operation

The SPIA interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDIA, SDOA, SCKA and $\overline{\text{SCSA}}$. Pins SDIA and SDOA are the Serial Data Input and Serial Data Output lines, the SCKA pin is the Serial Clock line and $\overline{\text{SCSA}}$ is the Slave Select line. As the SPIA interface pins are pin-shared with normal I/O pins, the SPIA interface must first be enabled by configuring the corresponding selection bits in the pin-shared function selection registers. The SPIA can be disabled or enabled using the SPIAEN bit in the SPIAC0 register. Communication between devices connected to the SPIA 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 each device only contains a single $\overline{SCSA}$ pin only one slave device can be utilized.

The $\overline{SCSA}$ pin is controlled by the application program, set the SACSEN bit to "1" to enable the $\overline{SCSA}$ pin function and clear the SACSEN bit to "0" to place the $\overline{SCSA}$ pin into a floating state.

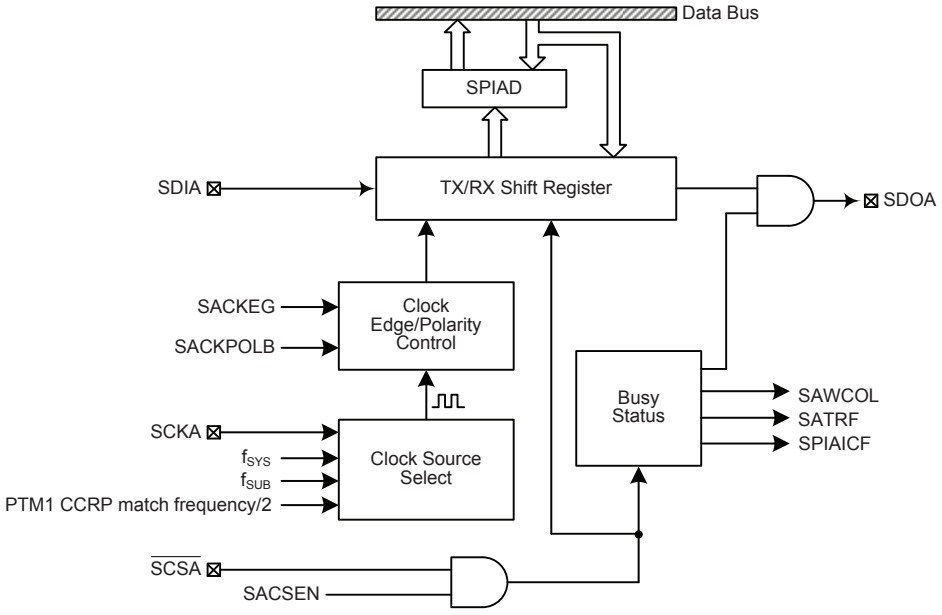


SPIA Master/Slave Connection

The SPIA function in these devices offer 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 SPIA 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 SACSEN and SPIAEN.



SPIA Block Diagram

SPIA Registers

There are three internal registers which control the overall operation of the SPIA interface. These are the SPIAD data register and two registers, SPIAC0 and SPIAC1.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SPIAC0	SASPI2	SASPI1	SASPI0	—	—	—	SPIAEN	SPIAICF
SPIAC1	—	—	SACKPOLB	SACKEG	SAMLS	SACSEN	SAWCOL	SATRF
SPIAD	D7	D6	D5	D4	D3	D2	D1	D0

SPIA Registers List

SPIA Data Register

The SPIAD register is used to store the data being transmitted and received. Before the device writes data to the SPIA bus, the actual data to be transmitted must be placed in the SPIAD register. After the data is received from the SPIA bus, the device can read it from the SPIAD register. Any transmission or reception of data from the SPIA bus must be made via the SPIAD register.

- **SPIAD 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**: SPIA Data Register bit 7~bit 0

SPIA Control Registers

There are also two control registers for the SPIA interface, SPIAC0 and SPIAC1. The SPIAC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SPIAC1 register is used for other control functions such as LSB/MSB selection, write collision flag etc.

- **SPIAC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SASPI2	SASPI1	SASPI0	—	—	—	SPIAEN	SPIAICF
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	1	1	1	—	—	—	0	0

Bit 7~5 **SASPI2~SASPI0**: SPIA Operating Mode Control

000: SPIA master mode; SPIA clock is $f_{SYS}/4$
001: SPIA master mode; SPIA clock is $f_{SYS}/16$
010: SPIA master mode; SPIA clock is $f_{SYS}/64$
011: SPIA master mode; SPIA clock is f_{SUB}
100: SPIA master mode; SPIA clock is PTM1 CCRP match frequency/2
101: SPIA slave mode
110: Unimplemented
111: Unimplemented

These bits are used to control the SPIA Master/Slave selection and the SPIA Master clock frequency. The SPIA clock is a function of the system clock but can also be chosen to be sourced from PTM1 and f_{SUB}. If the SPIA Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4~2 Unimplemented, read as "0"

Bit 1 **SPIAEN**: SPIA Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SPIA interface. When the SPIAEN bit is cleared to "0" to disable the SPIA interface, the SDIA, SDOA, SCKA and $\overline{SCSA}$ lines will lose their SPIA function and the SPIA operating current will be reduced to a minimum value. When the bit is high the SPIA interface is enabled.

Bit 0 **SPIAICF:** SPIA Incomplete Flag
0: SPIA incomplete condition is not occurred
1: SPIA incomplete condition is occured
This bit is only available when the SPIA is configured to operate in an SPIA slave mode. If the SPIA operates in the slave mode with the SPIAEN and SACSEN bits both being set to "1" but the $\overline{SCSA}$ line is pulled high by the external master device before the SPIA data transfer is completely finished, the SPIAICF bit will be set to 1 together with the SATRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the SATRF bit will not be set to "1" if the SPIAICF bit is set to "1" by software application program.

- **SPIAC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SACKPOLB	SACKEG	SAMLS	SACSEN	SAWCOL	SATRF
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **SACKPOLB**: SPIA Clock Line Base Condition Selection
0: The SCKA line will be high when the clock is inactive
1: The SCKA line will be low when the clock is inactive
The SACKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCKA line will be low when the clock is inactive. When the SACKPOLB bit is low, then the SCKA line will be high when the clock is inactive.

Bit 4 **SACKEG**: SPIA SCKA Clock Active Edge Type Selection
SACKPOLB=0
0: SCKA has high base level with data capture on SCKA rising edge
1: SCKA has high base level with data capture on SCKA falling edge
SACKPOLB=1
0: SCKA has low base level with data capture on SCKA falling edge
1: SCKA has low base level with data capture on SCKA rising edge
The SACKEG and SACKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPIA bus. These two bits must be configured before a data transfer is executed otherwise an erroneous clock edge may be generated. The SACKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCKA line will be low when the clock is inactive. When the SACKPOLB bit is low, then the SCKA line will be high when the clock is inactive. The SACKEG bit determines active clock edge type which depends upon the condition of the SACKPOLB bit.

Bit 3 **SAMLS**: SPIA 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 **SACSEN**: SPIA $\overline{SCSA}$ Pin Control
0: Disable
1: Enable

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

Bit 1 **SAWCOL**: SPIA Write Collision Flag
0: No collision
1: Collision

The SAWCOL 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 SPIAD 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 **SATRF**: SPIA Transmit/Receive Complete Flag
0: SPIA data is being transferred
1: SPIA data transmission is completed

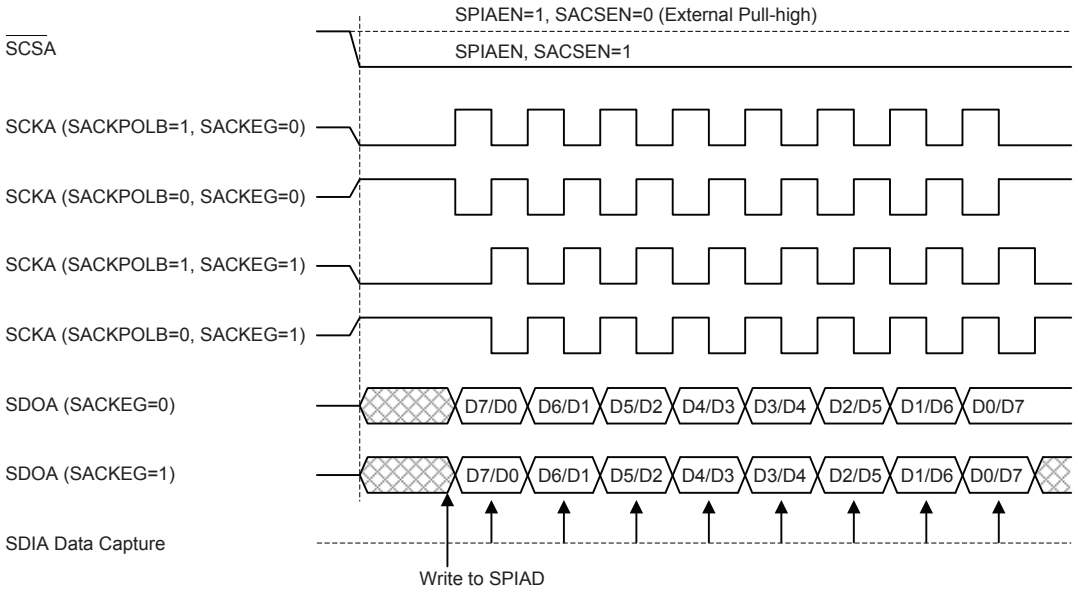
The SATRF bit is the Transmit/Receive Complete flag and is set "1" automatically when an SPIA data transmission is completed, but must set to "0" by the application program. It can be used to generate an interrupt.

SPIA Communication

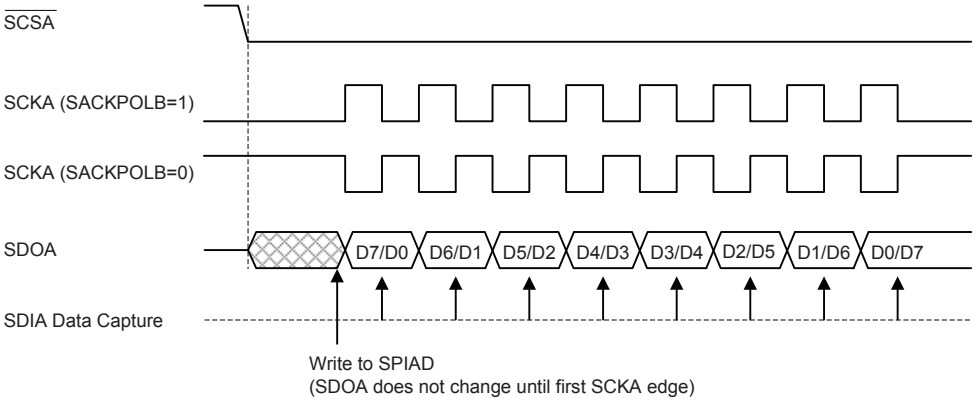
After the SPIA interface is enabled by setting the SPIAEN bit high, then in the Master Mode, when data is written to the SPIAD register, transmission/reception will begin simultaneously. When the data transfer is complete, the SATRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SPIAD register will be transmitted and any data on the SDIA pin will be shifted into the SPIAD register.

The master should output an $\overline{SCSA}$ signal to enable the slave device before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the $\overline{SCSA}$ signal depending upon the configurations of the SACKPOLB bit and SACKEG bit. The accompanying timing diagram shows the relationship between the slave data and $\overline{SCSA}$ signal for various configurations of the SACKPOLB and SACKEG bits.

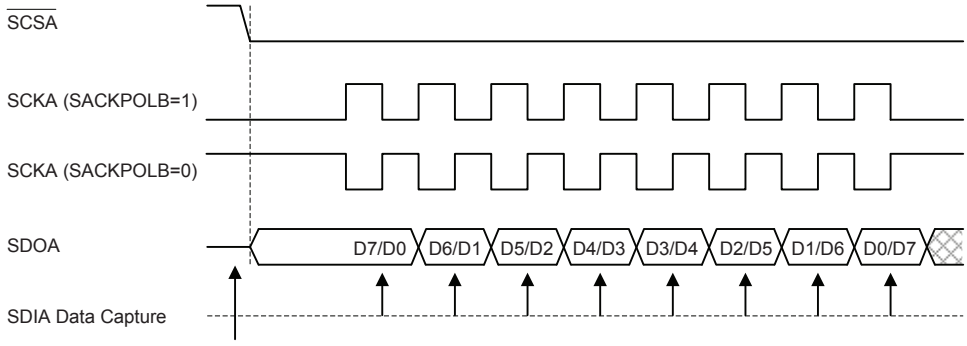
The SPIA will continue to function even in the IDLE Mode.



SPIA Master Mode Timing

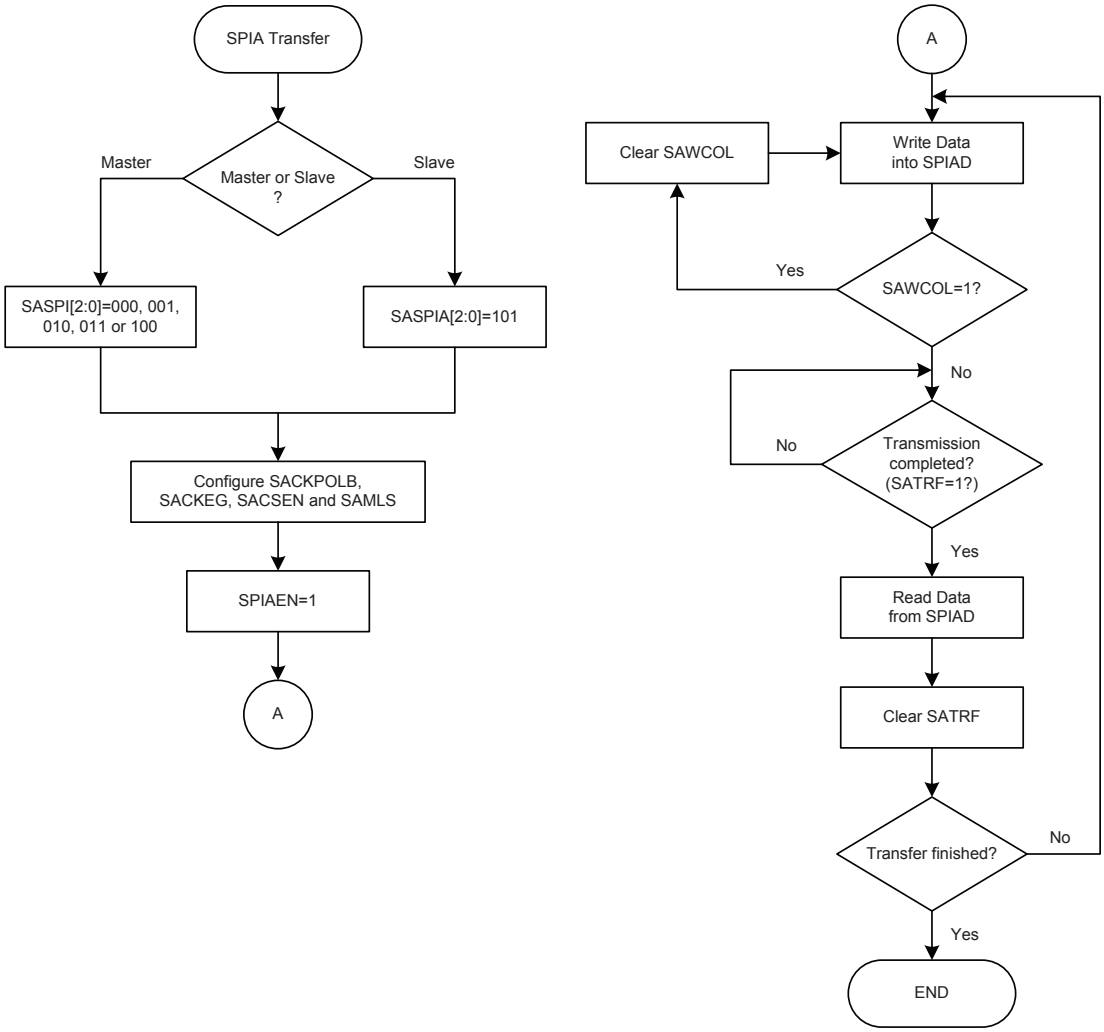


SPIA Slave Mode Timing – SACKEG=0



Note: For SPIA slave mode, if SPIAEN=1 and SACSEN=0, SPIA is always enabled and ignores the SCSA level.

SPIA Slave Mode Timing – SACKEG=1



SPIA Transfer Control Flowchart

SPIA Bus Enable/Disable

To enable the SPIA bus, set SACSEN=1 and $\overline{\text{SCSA}}$=0, then wait for data to be written into the SPIAD (TXRX buffer) register. For the Master Mode, after data has been written to the SPIAD (TXRX buffer) register, then transmission or reception will start automatically. When all the data has been transferred the SATRF bit should be set. For the Slave Mode, when clock pulses are received on SCKA, data in the TXRX buffer will be shifted out or data on SDIA will be shifted in.

To disable the SPIA bus SCKA, SDIA, SDOA, $\overline{\text{SCSA}}$ will become I/O pins or the other functions.

SPIA Operation

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

The SACSEN bit in the SPIAC1 register controls the overall function of the SPIA interface. Setting this bit high will enable the SPIA interface by allowing the $\overline{\text{SCSA}}$ line to be active, which can then be used to control the SPIA interface. If the SACSEN bit is low, the SPIA interface will be disabled and the $\overline{\text{SCSA}}$ line will be an I/O pin or the other functions and can therefore not be used for control of the SPIA interface. If the SACSEN bit and the SPIAEN bit in the SPIAC0 register are set high, this will place the SDIA line in a floating condition and the SDOA line high. If in Master Mode the SCKA line will be either high or low depending upon the clock polarity selection bit SACKPOLB in the SPIAC1 register. If in Slave Mode the SCKA line will be in a floating condition. If SPIAEN is low then the bus will be disabled and $\overline{\text{SCSA}}$, SDIA, SDOA and SCKA will all become I/O pins or the other functions. 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 SPIAD 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 clock source and Master mode using the SASPI2~SASPI0 bits in the SPIAC0 control register.
- Step 2
  Setup the SACSEN bit and setup the SAMLS bit to choose if the data is MSB or LSB first, this must be same as the Slave device.
- Step 3
  Setup the SPIAEN bit in the SPIAC0 control register to enable the SPIA interface.
- Step 4
  For write operations: write the data to the SPIAD register, which will actually place the data into the TXRX buffer. Then use the SCKA and $\overline{\text{SCSA}}$ lines to output the data. After this go to step 5.
  For read operations: the data transferred in on the SDIA line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPIAD register.
- Step 5
  Check the SAWCOL bit if set high then a collision error has occurred so return to step 4. If equal to “0” then go to the following step.
- Step 6
  Check the SATRF bit or wait for a SPIA serial bus interrupt.

- Step 7
  Read data from the SPIAD register.
- Step 8
  Clear SATRF.
- Step 9
  Go to step 4.

Slave Mode

- Step 1
  Select the SPIA Slave mode using the SASPI2~SASPI0 bits in the SPIAC0 control register.
- Step 2
  Setup the SACSEN bit and setup the SAMLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master device.
- Step 3
  Setup the SPIAEN bit in the SPIAC0 control register to enable the SPIA interface.
- Step 4
  For write operations: write the data to the SPIAD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCKA and $\overline{\text{SCSA}}$ signal. After this, go to step 5.
  For read operations: the data transferred in on the SDIA line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPIAD register.
- Step 5
  Check the SAWCOL bit if set high then a collision error has occurred so return to step 4. If equal to "0" then go to the following step.
- Step 6
  Check the SATRF bit or wait for a SPIA serial bus interrupt.
- Step 7
  Read data from the SPIAD register.
- Step 8
  Clear SATRF.
- Step 9
  Go to step 4.

Error Detection

The SAWCOL bit in the SPIAC1 register is provided to indicate errors during data transfer. The bit is set by the SPIA serial Interface but must be cleared by the application program. This bit indicates a data collision has occurred which happens if a write to the SPIAD register takes place during a data transfer operation and will prevent the write operation from continuing.

UART Interface

Each 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 possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UART function contains the following features:

- Full-duplex, Universal Asynchronous Receiver and Transmitter (UART) 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)
- Transmitter and receiver enabled independently
- 2-byte deep FIFO receive data buffer
- Transmit and receive multiple interrupt generation sources
  - Transmitter empty
  - Transmitter idle
  - Receiver full
  - Receiver overrun
  - Address mode detect
  - RX pin wake-up interrupt (RX enable, RX falling edge)

UART External Pin Interfacing

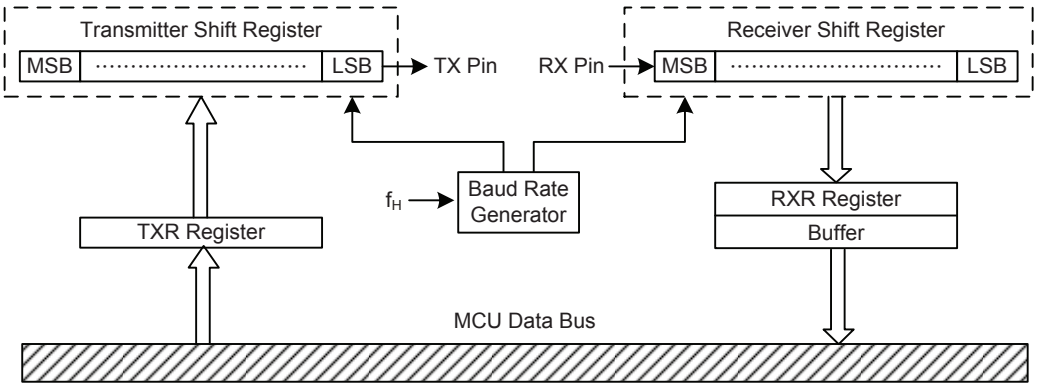
To communicate with an external serial interface, the internal UART has two external pins known as TX and RX. The TX and RX pins are the UART transmitter and receiver pins respectively. Along with the UARTEN bit, the TXEN and RXEN bits, if set, will automatically setup these I/O or other pin-shared functional pins to their respective TX output and RX input conditions and disable any pull-high resistor option which may exist on the TX or RX pins. When the TX or RX pin function is disabled by clearing the UARTEN and TXEN or RXEN bit, the TX or RX pin can be used as a general purpose I/O or other pin-shared functional pin.

UART Data Transfer Scheme

The block diagram shows the overall data transfer structure arrangement for the UART interface. The actual data to be transmitted from the MCU is first transferred to the TXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate Generator. Only the TXR 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 RXR register, where it is buffered and can be manipulated by the application program. Only the 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, although referred to in the text, and in application programs, as separate TXR and RXR registers, only exists as a single shared register in the Data Memory. This shared register known as the TXR_RXR register is used for both data transmission and data reception.



UART Data Transfer Scheme

UART Status and Control Registers

There are five control registers associated with the UART function. The USR, UCR1 and UCR2 registers control the overall function of the UART, while the BRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXR data register.

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

UART Register List

USR Register

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

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

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

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

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

Bit 5 **FERR**: Framing Error Flag
0: No framing error is detected
1: Framing error is detected
The FERR flag is the framing error flag. When this read only flag is "0", it indicates that there is no framing error. When the flag is "1", it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register USR followed by an access to the RXR data register.

Bit 4 **OERR**: Overrun Error Flag
0: No overrun error is detected
1: Overrun error is detected
The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is "0", it indicates that there is no overrun error. When the flag is "1", it indicates that an overrun error occurs which will inhibit further transfers to the RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register USR followed by an access to the RXR data register.

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

Bit 2 **RXIF**: Receive RXR Data Register Status
0: RXR data register is empty
1: RXR data register has available data
The RXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the RXR read data register is empty. When the flag is "1", it indicates that the RXR read data register contains new data. When the contents of the shift register are transferred to the RXR register, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag is cleared when the USR register is read with RXIF set, followed by a read from the RXR register, and if the RXR register has no data available.

Bit 1 **TIDLE**: Transmission Idle
0: Data transmission is in progress (data being transmitted)
1: No data transmission is in progress (transmitter is idle)
The TIDLE flag is known as the transmission complete flag. When this read only flag is "0", it indicates that a transmission is in progress. This flag will be set to "1" when the TXIF flag is "1" and when there is no transmit data or break character being transmitted. When TIDLE is equal to "1", the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared by reading the USR register with TIDLE set and then writing to the TXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 **TXIF**: Transmit TXR Data Register Status
0: Character is not transferred to the transmit shift register
1: Character has transferred to the transmit shift register (TXR data register is empty)

The TXIF flag is the transmit data register empty flag. When this read only flag is "0", it indicates that the character is not transferred to the transmitter shift register. When the flag is "1", it indicates that the transmitter shift register has received a character from the TXR data register. The TXIF flag is cleared by reading the UART status register (USR) with TXIF set and then writing to the TXR data register. Note that when the TXEN bit is set, the TXIF flag bit will also be set since the transmit data register is not yet full.

UCR1 Register

The UCR1 register together with the UCR2 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	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

"x" unknown

Bit 7 **UARTEN**: UART Function Enable Control
0: Disable UART. TX and RX pins are in a floating state
1: Enable UART. TX and RX pins function as UART pins

The UARTEN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX 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 and the TX and RX pins will function as defined by the TXEN and RXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2 and BRG registers will remain unaffected. If the UART is active and the UARTEN 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 **BNO**: Number of Data Transfer Bits Selection
0: 8-bit data transfer
1: 9-bit data transfer

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

Note: If BNO=1 (9-bit data transfer), parity function is enabled, the 9th bit of data is the parity bit which will not be transferred to RX8.
If BNO=0 (8-bit data transfer), parity function is enabled, the 8th bit of data is the parity bit which will not be transferred to RX7.

Bit 5 **PREN**: Parity Function Enable Control
0: Parity function is disabled
1: Parity function is enabled

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

Bit 4 **PRT**: Parity Type Selection Bit
0: Even parity for parity generator
1: Odd parity for parity generator
This bit is the parity type selection bit. When this bit is equal to "1", odd parity type will be selected. If the bit is equal to "0", then even parity type will be selected.

Bit 3 **STOPS**: Number of Stop Bits Selection
0: One stop bit format is used
1: Two stop bits format is used
This bit determines if one or two stop bits are to be used for the TX pin. When this bit is equal to "1", two stop bits are used. If this bit is equal to "0", then only one stop bit is used.

Bit 2 **TXBRK**: Transmit Break Character
0: No break character is transmitted
1: Break characters transmit
The TXBRK bit is the Transmit Break Character bit. When this bit is "0", there are no break characters and the TX pin operates normally. When the bit is "1", there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to "1", after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.

Bit 1 **RX8**: Receive Data Bit 8 for 9-bit Data Transfer Format (read only)
This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **TX8**: Transmit Data Bit 8 for 9-bit Data Transfer Format (write only)
This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

UCR2 register

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

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

Bit 7 **TXEN**: UART Transmitter Enable Control
0: UART transmitter is disabled
1: UART transmitter is enabled
The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to "0", the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TX pin will be set in a floating state.
If the TXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be set in a floating state.

Bit 6 **RXEN**: UART Receiver Enable Control
0: UART receiver is disabled
1: UART receiver is enabled
The bit named RXEN is the Receiver Enable Bit. When this bit is equal to "0", the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RX pin will be set in a floating state. If the RXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the receiver will be enabled and the RX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX pin will be set in a floating state.

Bit 5 **BRGH**: Baud Rate Speed Selection
0: Low speed baud rate
1: High speed baud rate
The bit named BRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register BRG, controls the Baud Rate of the UART. If this bit is equal to "1", the high speed mode is selected. If the bit is equal to "0", the low speed mode is selected.

Bit 4 **ADDEN**: Address Detect Function Enable Control
0: Address detection function is disabled
1: Address detection function is enabled
The bit named ADDEN is the address detect function enable control bit. When this bit is equal to "1", the address detect function is enabled. When it occurs, if the 8^{th} bit, which corresponds to RX7 if BNO=0 or the 9^{th} bit, which corresponds to RX8 if BNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8^{th} or 9^{th} bit depending on the value of BNO. If the address bit known as the 8^{th} or 9^{th} bit of the received word is "0" with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **WAKE**: RX Pin Falling Edge Wake-up UART Function Enable Control
0: RX pin wake-up UART function is disabled
1: RX pin wake-up UART function is enabled
This bit is used to control the wake-up UART function when a falling edge on the RX pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no RX pin wake-up UART function if the UART clock (f_H) exists. If the WAKE bit is set to 1 as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the RX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function can not resume even if there is a falling edge on the RX pin when the WAKE bit is cleared to "0".

Bit 2 **RIE**: Receiver Interrupt Enable Control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled
This bit enables or disables the receiver interrupt. If this bit is equal to "1" and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.

Bit 1 **TIIE**: Transmitter IdleInterrupt Enable Control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled
This bit enables or disables the transmitter idle interrupt. If this bit is equal to "1" and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

Bit 0 **TEIE**: Transmitter Empty Interrupt Enable Control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled
This bit enables or disables the transmitter empty interrupt. If this bit is equal to "1" and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

TXR_RXR Register

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

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

"x" unknown

Bit 7~0 **D7~D0:** UART Transmit/Receive Data bit 7~bit 0

Baud Rate Generator

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

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

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

- **BRG Register**

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

"x" unknown

Bit 7~0 **D7~D0**: Baud RateValues

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

Calculating the Baud Rate and error values

For a clock frequency of 4MHz, and with BRGH set to "0" determine the BRG register value N, the actual baud rate and the error value for a desired baud rate of 4800.

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

Re-arranging this equation gives N = [f_H / (BR × 64)] − 1

Giving a value for N = [4000000 / (4800×64)] − 1 = 12.0208

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

Therefore the error is equal to (4808 − 4800) / 4800 = 0.16%

The following table shows actual values of baud rate and error values for the two values of BRGH.

Baud Rate K/BPS	f_{SYS}=8MHz					
	Baud Rates for BRGH=0			Baud Rates for BRGH=1		
	BRG	Kbaud	Error (%)	BRG	Kbaud	Error (%)
0.3	—	—	—	—	—	—
1.2	103	1.202	0.16	—	—	—
2.4	51	2.404	0.16	207	2.404	0.16
4.8	25	4.808	0.16	103	4.808	0.16
9.6	12	9.615	0.16	51	9.615	0.16
19.2	6	17.8857	-6.99	25	19.231	0.16
38.4	2	41.667	8.51	12	38.462	0.16
57.6	1	62.500	8.51	8	55.556	-3.55
115.2	0	125	8.51	3	125	8.51
250	—	—	—	1	250	0

Baud Rates and Error Values

UART Setup and Control

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

Enabling/Disabling the UART Interface

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

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

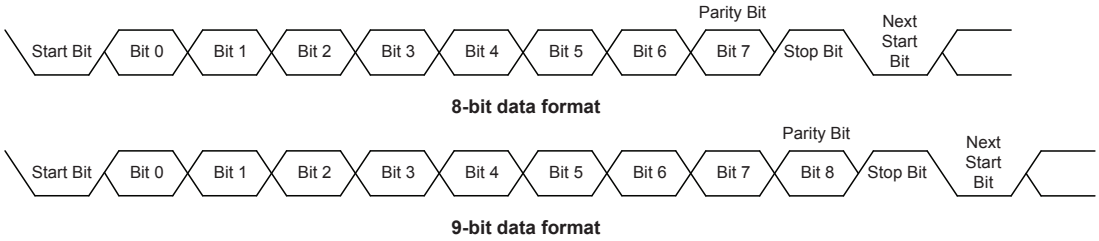
Data, Parity and Stop Bit Selection

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

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

Transmitter Receiver Data Format

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



UART Transmitter

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

Transmitting Data

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

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

This sequence of events can now be repeated to send additional data. It should be noted that when TXIF is "0", data will be inhibited from being written to the TXR register. Clearing the TXIF flag is always achieved using the following software sequence:

- A USR register access
- A TXR register write execution

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

- A USR register access
- A TXR register write execution

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

Transmit Break

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

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR register. If the BNO bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8 bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX 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 to the shift register, with the least significant bit LSB first. The 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 RXR before the third byte has been completely shifted in, otherwise this third byte will be discarded and an overrun error OERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNO, PRT and PREN bits to define the word length and parity type.
- Setup the BRG register to select the desired baud rate.
- Set the RXEN bit to ensure that the UART receiver is enabled and 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 RXIF bit in the USR register will be set when the RXR register has data available. There will be at most one more characters available before an overrun error occurs.
- When the contents of the shift register have been transferred to the RXR register and if the RIE bit is set, then an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

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

- A USR register access
- An 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 BNO and plusing one STOP bit. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO and plusing one STOP bit. The RXIF bit is set, FERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLE bit is set. A break is regarded as a character that contains only zeros with the FERR flag set. If a long break signal has been detected, the receiver will regard it as a data frame including a start bit, data bits and the invalid stop bit and the FERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

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

Idle Status

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

Receiver Interrupt

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

Managing Receiver Errors

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

Overrun Error – OERR flag

The 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 RXR register. If this is not done, the overrun error flag OERR will be consequently indicated.

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

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

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

Noise Error – NF Flag

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

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

Note that the NF flag is reset by a USR register read operation followed by an RXR register read operation.

Framing Error – FERR Flag

The read only framing error flag, FERR, in the USR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, only the first stop bit is detected, it must be high. If the first stop bit is low, the FERR flag will be set. The FERR flag and the received data will be recorded in the USR and RXR registers respectively, and the flag is cleared in any reset.

Parity Error – PERR Flag

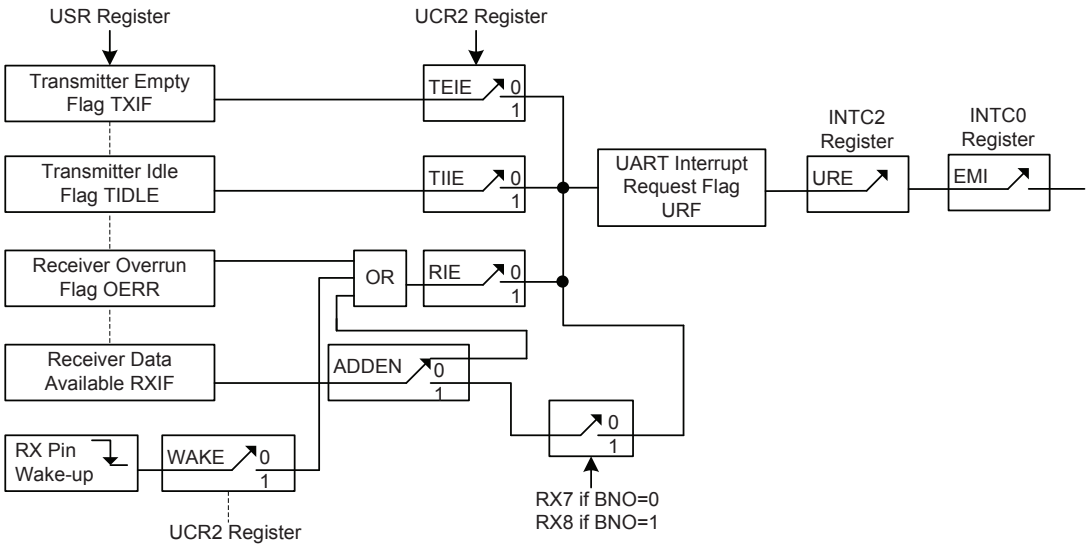
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN bit is "1", and if the parity type, odd or even is selected. The read only PERR flag and the received data will be recorded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

UART Module Interrupt Structure

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

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

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



UART Interrupt Scheme

Address Detect Mode

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is "1", then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the related interrupt enable control bit and the EMI bit must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO bit is "1" or the 8th bit if BNO bit is "0". If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is "0", then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to "0".

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

ADDEN Bit Function

UART Power Down and Wake-up

When the UART clock (f_H) is switched off, the UART will cease to function, 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 (f_H) source derived from the microcontroller is activated. In a similar way, if the device executes the "HALT" instruction and switches off the system clock while receiving data, then the reception of data will likewise be paused. When the device enters the IDLE or SLEEP Mode, note that the USR, UCR1, UCR2, transmit and receive registers, as well as the BRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the 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 UART interrupt enable bit, URE, must also be set. If these two bits are not set then only a wake up event will occur and no interrupt will be generated. Note also that as it takes certain system clock cycles after a wake-up before normal microcontroller resumes, the UART interrupt will not be generated until after this time has elapsed.

Body Fat Measurement Function

The body fat circuit consists of sine generator, operational amplifier and filter. It is high quality, flexibility and high integration for body fat measurement. The whole module power is from LDO.

Sine Wave Generator

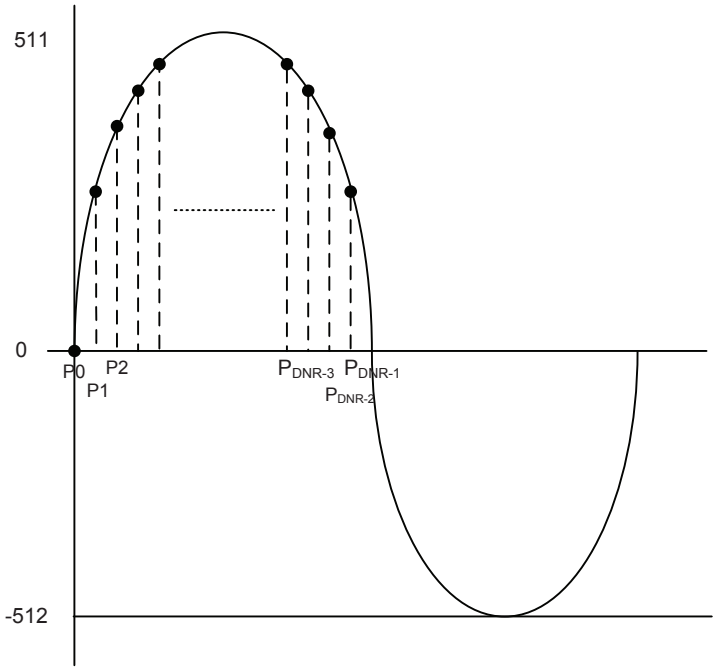
The sine generator consists of a frequency divider, counter, RAM, 10-bit D/A converter and OP0. It offers the wide range 5kHz~500kHz sine wave generator and 32×9 bits RAM for sine wave pattern by software setting.The frequency divider will multiply by DN/M to generate a clock to counter. The related details refer to following formula.

- System clock / M = sine wave frequency
- System clock × (DN/M) = counter count rate
- The M must be the multiple of N and 8.
- M = N × DN
- DNR = DN / 2
- DN: sine wave cycle data number (DN ≤ 64)
- DNR: 1/2 sine wave cycle stored in RAM(DNR ≤ 32) data number

Please refer to following table and figure in details.

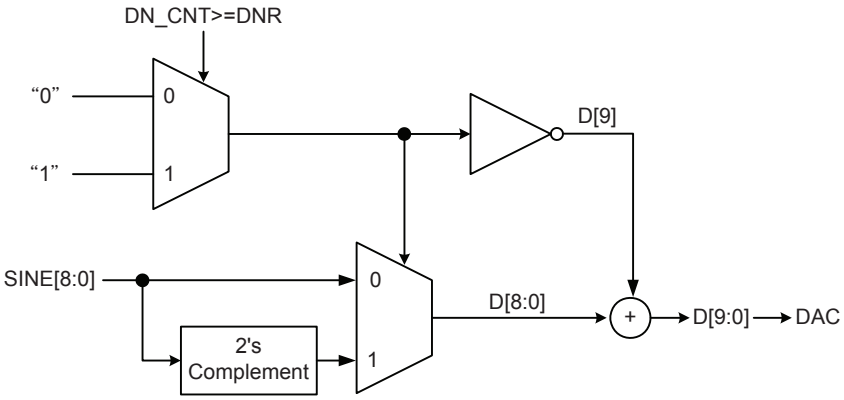
System Frequency	4MHz			8MHz			12MHz		
The Frequency of Sine Wave (kHz)	**500**	**50**	**5**	**500**	**50**	**5**	**500**	**50**	**5**
M	8	80	800	16	160	1600	24	240	2400
N	1	2	20	1	4	25	1	5	50
DN	8	40	40	16	40	64	24	48	48
DNR	4	20	20	8	20	32	12	24	24

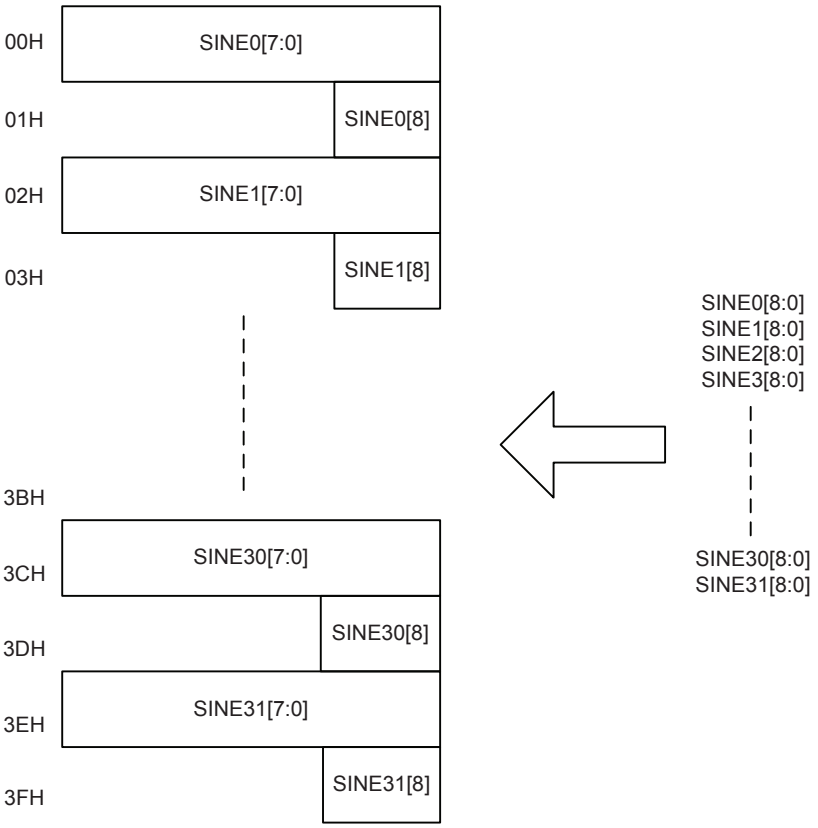
Note: The sine wave generator circuit consists of a 10-bit D/A converter and a smoothing filter whose frequency at -3dB is equal to 489kHz. When the output frequency is 500kHz, the output wave amplitude will decrease and therefore it is impossible to achieve a full range of V_{OREG}~0.



It is only necessary to generate a half sine wave pattern P0~PDNR-1 which is stored in RAM sector 3 (00H~3FH). The sine pattern [7:0] is stored using even addresses and the sine pattern [8] is stored using odd addresses. Once the sine generator is enabled, the CPU cannot writes/read any data to/from this RAM. The sine generator will then read this RAM data and transmit it to an 10-bit D/A converter.

The controller reads a half sine pattern from RAM and generates a sine waveform on the SIN pin. Refer to following figure.





RAM (Sector 3) **SINE Wave Pattern**

Body Fat Measurement Registers

There are a series of registers control the overall operation of the body fat measurement function.

Sine Wave Generator

The sine wave generator is controlled by three registers. Details are given as below.

- **SGC Register**

Bit	7	6	5	4	3	2	1	0
Name	SGEN	D6	D5	BREN	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

Bit 7 **SGEN**: Sine Generator Enabled
0: Disable
1: Enable
When this bit is equal to "0", the OP0 and 10-bit D/A converter will be in a power down mode.

Bit 6~5 **D6~D5**: Reserved bits, must be fixed as "00".

Bit 4 **BREN**: Bias Resistors Control bit
0: Disable – power down mode
1: Enable – normal mode

Bit 3~0 Unimplemented, read as "0"

- **SGN Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **D5~D0**: Sine Generator Data
The multiplicator of system frequency (N)
The multiplicator (N) is equal to D[5:0]+1

- **SGDNR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	D4	D3	D2	D1	D0
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 **D4~D0**: Data Number of Sample
1/2 sine wave cycle numerical value is stored in RAM Sector 3. DNR is equal to D[4:0]+1.

Amplifier

Five registers are associated with the amplifier operation. The OPAC register for controlling the operational amplifier, the SWC0, SWC1 and SWC2 registers for configuring the switch condition as well as the DACO register for setting the 6-bit D/A converter output.

- **OPAC Register**

Bit	7	6	5	4	3	2	1	0
Name	OPAEN	—	—	—	OP2G3	OP2G2	OP2G1	OP2G0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **OPAEN**: Operational Amplifier Control
0: Disable
1: Enable
When this bit is equal to "0", OP1, OP2 and 6-bit D/A converter will be in a power down mode.

Bit 6~4 Unimplemented, read as "0"

Bit 3~0 **OP2G3~OP2G0**: OP2 Gain Control
0001: 1.14
0010: 1.31
0011: 1.50
0100: 1.73
0101: 2.00
0110: 2.33
0111: 2.75
1000: 3.285
1001: 4.00
1010: 5.00
others: 1.00

- **SWC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	IHR2	VHR1	IHL1	VHL2	IFR2	VFR1	IFL1	VFL2
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 **IHR2**: Switch IHR2 Control Bit
0: Off
1: On

Bit 6 **VHR1**: Switch VHR1 Control Bit
0: Off
1: On

Bit 5 **IHL1**: Switch IHL1 Control Bit
0: Off
1: On

Bit 4 **VHL2**: Switch VHL2 Control Bit
0: Off
1: On

Bit 3 **IFR2**: Switch IFR2 Control Bit
0: Off
1: On

Bit 2 **VFR1**: Switch VFR1 Control Bit
0: Off
1: On

Bit 1 **IFL1**: Switch IFL1 Control Bit
0: Off
1: On

Bit 0 **VFL2**: Switch VFL2 Control Bit
0: Off
1: On

- **SWC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	VHL1	IHL2	VRFN	VRFP1	IRF1	VRFP2	IRF2
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 **VHL1**: Switch VHL1 Control Bit
0: Off
1: On

Bit 5 **IHL2**: Switch IHL1 Control Bit
0: Off
1: On

Bit 4 **VRFN**: Switch VRFN Control Bit
0: Off
1: On

Bit 3 **VRFP1**: Switch VRFP1 Control Bit
0: Off
1: On

Bit 2 **IRF1**: Switch IRF1 Control Bit
0: Off
1: On

Bit 1 **VRFP2**: Switch VRFP2 Control Bit
0: Off
1: On

Bit 0 **IRF2**: Switch IRF2 Control Bit
0: Off
1: On

- **SWC2 register**

Bit	7	6	5	4	3	2	1	0
Name	—	VHR2	IHR1	VFL1	VFR2	IFL2	IFR1	—
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	—	0	0	0	0	0	0	—

Bit 7 Unimplemented, read as "0"

Bit 6 **VHR2**: switch VHR2 control bit
0: Off
1: On

Bit 5 **IHR1**: switch IHR1 control bit
0: Off
1: On

Bit 4 **VFL1**: switch VFL1 control bit
0: Off
1: On

Bit 3 **VFR2**: switch VFR2 control bit
0: Off
1: On

Bit 2 **IFL2**: switch IFL2 control bit
0: Off
1: On

Bit 1 **IFR1**: switch IFR1 control bit
0: Off
1: On

Bit 0 Unimplemented, read as "0"

- **DACO Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **D5~D0**: 6-bit D/A Converter Output Voltage=$0.5V_{OREG} \times ((D[5:0]+1)/64)$

Filter

The FTRC register controls the operation of the filter part.

- **FTRC Register**

Bit	7	6	5	4	3	2	1	0
Name	FTREN	—	—	HYSEN	—	—	SW19	SW18
R/W	R/W	—	—	R/W	—	—	R/W	R/W
POR	0	—	—	0	—	—	0	0

Bit 7 **FTREN**: Filter Control Bit
0: Disable
1: Enable
When this bit is equal to "0", CP0 and PMOS will be in a power down mode.

Bit 6~5 Unimplemented, read as "0"

Bit 4 **HYSEN**: CP0 Hysteresis control bit
0: Disable
1: Enable

Bit 3~2 Unimplemented, read as "0"

Bit 1 **SW19**: Switch 19 Control Bit
0: Off
1: On

Bit 0 **SW18**: Switch 18 Control Bit
0: Off
1: On

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 devices contain several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the Timer Modules (TMs), Time Bases, Serial Interface Module (SIM), Serial Peripheral Interface (SPIA), UART, Low Voltage Detector (LVD), EEPROM and the A/D converter.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory. The registers fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI2 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
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~2
Time Base	TBnE	TBnF	n=0~1
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
UART	URE	URF	—
SIM	SIME	SIMF	—
SPIA	SPIAE	SPIAF	—
STM	STMPE	STMPF	—
	STMAE	STMAF	
PTM	PTMnPE	PTMnPF	n=0~2
	PTMnAE	PTMnAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
INTC1	—	MF2F	MF1F	MF0F	—	MF2E	MF1E	MF0E
INTC2	SIMF	URF	TB1F	TB0F	SIME	URE	TB1E	TB0E
INTC3	—	—	—	SPIAF	—	—	—	SPIAE
MFI0	PTM0AF	PTM0PF	STMAF	STMPF	PTM0AE	PTM0PE	STMAE	STMPE
MFI1	PTM2AF	PTM2PF	PTM1AF	PTM1PF	PTM2AE	PTM2PE	PTM1AE	PTM1PE
MFI2	—	—	DEF	LVF	—	—	DEE	LVE

Interrupt Register List

INTEG Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INT1S1~INT1S0**: Interrupt Edge Control for INT1 Pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt Edge Control for INT0 Pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **ADF**: A/D Converter Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **INT1F**: External Interrupt 1Request Flag
0: No request
1: Interrupt request

Bit 4 **INT0F**: External Interrupt 0 Request Flag
0: No request
1: Interrupt request

Bit 3 **ADE**: A/D Converter 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	—	MF2F	MF1F	MF0F	—	MF2E	MF1E	MF0E
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **MF2F**: Multi-function Interrupt 2 Request Flag
0: No request
1: Interrupt request

Bit 5 **MF1F**: Multi-function Interrupt 1 Request Flag
0: No request
1: Interrupt request

Bit 4 **MF0F**: Multi-function Interrupt 0 Request Flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **MF2E**: Multi-function Interrupt 2 Control
0: Disable
1: Enable

Bit 1 **MF1E**: Multi-function Interrupt 1 Control
0: Disable
1: Enable

Bit 0 **MF0E**: Multi-function Interrupt 0 Control
0: Disable
1: Enable

INTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	SIMF	URF	TB1F	TB0F	SIME	URE	TB1E	TB0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **SIMF**: Serial Interface Module Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **URF**: UART Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **TB1F**: Time Base 1 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **TB0F**: Time Base 0 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **SIME**: Serial Interface Module Interrupt Control
0: Disable
1: Enable

Bit 2 **URE**: Serial Interface Module Interrupt Control
0: Disable
1: Enable

Bit 1 **TB1E**: Time Base 1 Interrupt Control
0: Disable
1: Enable

Bit 0 **TB0E**: Time Base 0 Interrupt Control
0: Disable
1: Enable

INTC3 Register

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

Bit 7~5 Unimplemented, read as "0"

Bit 4 **SPIAF**: SPIA Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3~1 Unimplemented, read as "0"

Bit 0 **SPIAE**: SPIA Interrupt Control
0: Disable
1: Enable

MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	PTM0AF	PTM0PF	STMAF	STMPF	PTM0AE	PTM0PE	STMAE	STMPE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PTM0AF**: PTM0 CCRA Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **PTM0PF**: PTM0 CCRP Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **STMAF**: STM CCRA Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **STMPF**: STM CCRP Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **PTM0AE**: PTM0 CCRA Comparator Interrupt Control
0: Disable
1: Enable

Bit 2 **PTM0PE**: PTM0 CCRP Comparator Interrupt Control
0: Disable
1: Enable

Bit 1 **STMAE**: STM CCRA Comparator Interrupt Control
0: Disable
1: Enable

Bit 0 **STMPE**: STM CCRP Comparator Interrupt Control
0: Disable
1: Enable

MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	PTM2AF	PTM2PF	PTM1AF	PTM1PF	PTM2AE	PTM2PE	PTM1AE	PTM1PE
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 **PTM2AF:** PTM2 CCRA Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **PTM2PF**: PTM2 CCRP Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **PTM1AF**: PTM1 CCRA Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **PTM1PF**: PTM1 CCRP Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **PTM2AE**: PTM2 CCRA Comparator Interrupt Control
0: Disable
1: Enable

Bit 2 **PTM2PE**: PTM2 CCRP Comparator Interrupt Control
0: Disable
1: Enable

Bit 1 **PTM1AE**: PTM1 CCRA Comparator Interrupt Control
0: Disable
1: Enable

Bit 0 **PTM1PE**: PTM1 CCRP Comparator Interrupt Control
0: Disable
1: Enable

MFI2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	DEF	LVF	—	—	DEE	LVE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **DEF**: Data EEPROM Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **LVF**: LVD Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **DEE**: Data EEPROM Interrupt Control
0: Disable
1: Enable

Bit 0 **LVE**: LVD Interrupt Control
0: Disable
1: Enable

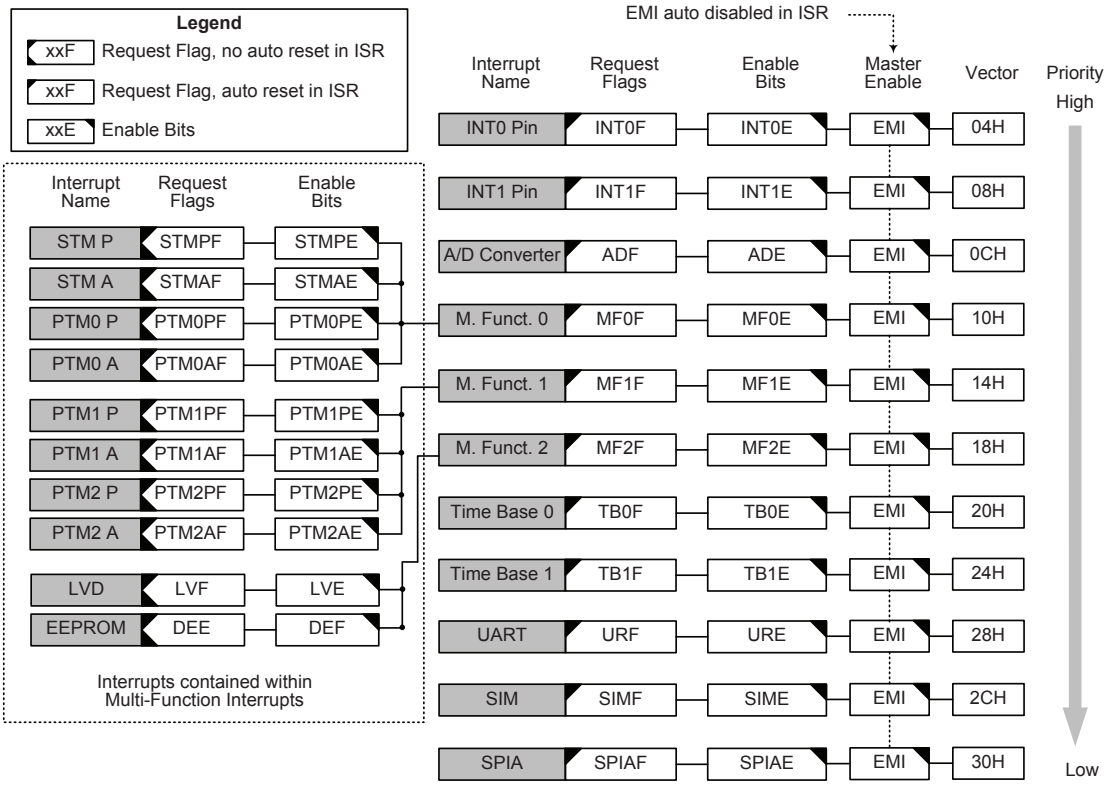
Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P, Comparator A match or A/D conversion completion etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is "0" then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

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

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. 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.



Interrupt Structure

External Interrupt

The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register.

When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input. The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

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.

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.

A/D Converter Interrupt

The A/D Converter Interrupt is controlled by the termination of an A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter Interrupt vector, will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Multi-function Interrupts

Within these devices there are three Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupts, EEPROM interrupt and LVD interrupt.

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

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

Serial Interface Module Interrupt

The Serial Interface Module Interrupt is also known as the SIM Interrupt. A SIM Interrupt request will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface, an I^2C address match or I^2C time-out occurrence. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, SIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the SIM interrupt vector, will take place. When the SIM Interface Interrupt is serviced, the interrupt request flag, SIMF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

SPIA Interrupt

The Serial Peripheral Interface Interrupt, also known as the SPIA Interrupt, will take place when the SPIA Interrupt request flag, SPIAF, is set, which occurs when a byte of data has been received or transmitted by the SPIA interface. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, SPIAE, must first be set. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SPIA interface, a subroutine call to the respective Interrupt vector, will take place. When the interrupt is serviced, the Serial Interface Interrupt flag, SPIAF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

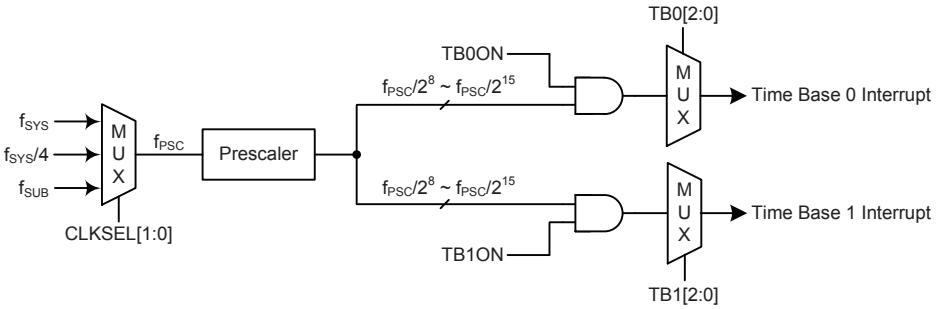
UART Interrupt

Several individual UART conditions can generate a UART Interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. To allow the program to branch to the respective interrupt vector addresses, the global interrupt enable bit, EMI, and UART Interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of these conditions are created, a subroutine call to the UART Interrupt vector will take place. When the interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART section.

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 Interrupt

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

Timer Module Interrupts

Each of the Standard Type TM and Periodic Type TM has two interrupts. All of the TM interrupts are contained within the Multi-function Interrupts. For the Standard Type TM and the Periodic Type TM, each has two interrupt request flags of STMPF, STMAF and PTMnPF, PTMnAF and two enable bits of STMPE, STMAE and PTMnPE, PTMnAE. 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, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector 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 MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

Interrupt Wake-up Function

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

Programming Considerations

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

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MFnF, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

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

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

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

Low Voltage Detector – LVD

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

LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of eight fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} or LVDIN pin input voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to "0" 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 Detect
1: Low Voltage Detect

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

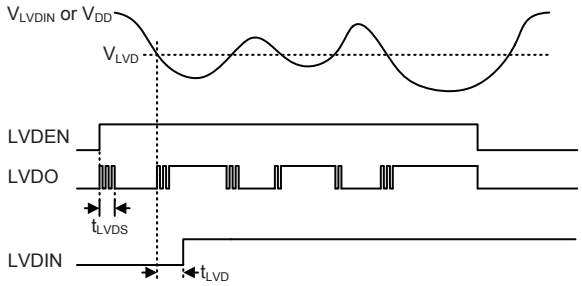
Bit 3 **VBGEN**: Bandgap Buffer Control
0: Disable
1: Enable

Bit 2~0 **VLVD2~VLVD0**: Select LVD Voltage
000: $V_{LVDIN} \leq 1.04V$
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

Note: When the VLVD bit field is set to 000B, the LVD function will be implemented by comparing the LVD reference voltage with a voltage value of 1.04V which is derived from the LVDIN pin. Otherwise, the LVD function will operate by comparing the LVD reference voltage with a specific voltage value which is generated by the internal LVD circuit when the VLVD bit field is set to any other value except 000B.

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD} or LVDIN pin input voltage with a pre-specified voltage level stored in the LVDC register. When the power supply voltage, V_{DD} or LVDIN pin input voltage fall below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is in the SLEEP mode, the low voltage detector will be disabled even if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage or LVDIN pin pinput voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



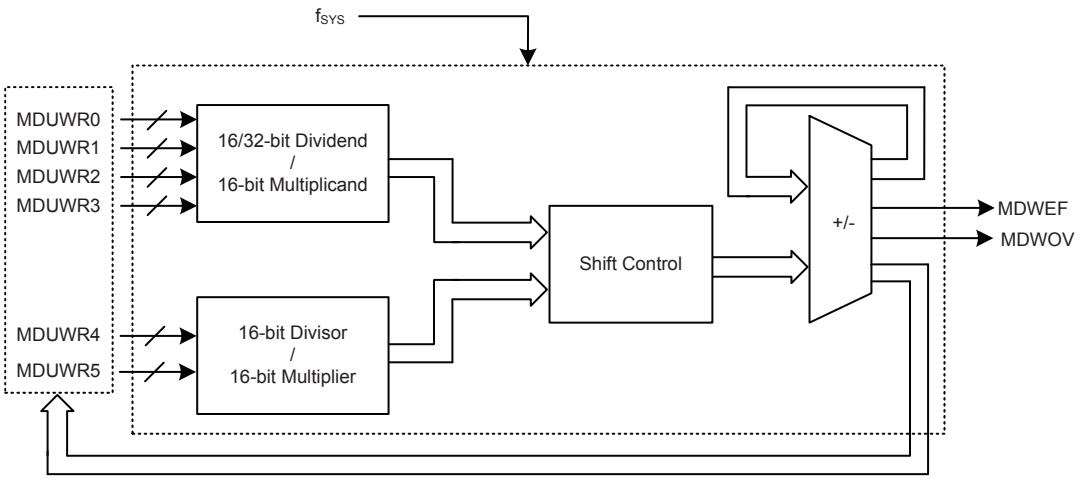
LVD Operation

The Low Voltage Detector interrupt is contained within the Multi-function interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. 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} or LVDIN pin input voltage falls below the preset LVD voltage. This will cause the device to wake-up from the IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before the device enters the IDLE Mode.

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

16-bit Multiplication Division Unit – MDU

Each device contains a 16-bit Multiplication Division Unit, MDU, which integrates a 16-bit unsigned multiplier and a 32-bit/16-bit divider. The MDU, in replacing the software multiplication and division operations, can therefore save large amounts of computing time as well as the Program and Data Memory space. It also reduces the overall microcontroller loading the results in the overall system perfofmance improvements.



16-bit MDU Block Diagram

MDU registers

The multiplication and division operations are implemented in a specific way, a specific write access sequence of a series of MDU data registers. The status register, MDUWCTRL, provides the indications for the MDU operation. The data register each is used to store the data regarded as the different operand corresponding to different MDU operations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
MDUWR0	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR1	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR2	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR3	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR4	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR5	D7	D6	D5	D4	D3	D2	D1	D0
MDUWCTRL	MDWEF	MDWOV	–	–	–	–	–	–

MDU Registers List

MDUWRn Register (n=0~5)

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

"x" unknown

Bit 7~0 **D7~D0**: 16-bit MDU data register n

MDUWCTRL Register

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

Bit 7 **MDWEF**: 16-bit MDU Error Flag
0: Normal
1: Abnormal
This bit will be set to 1 if the data register MDUWRn is written or read as the MDU operation is executing. This bit should be cleared to 0 by reading the MDUWCTRL register if it is equal to 1 and the MDU operation is completed.

Bit 6 **MDWOV**: 16-bit MDU Overflow Flag
0: No overflow occurs
1: Multiplication product > FFFFH or Divisor = 0
When an operation is completed, this bit will be updated by hardware to a new value corresponding to the current operation situation.

Bit 5~0 Unimplemented, read as "0"

Multiplication Division Unit Operation

For this MDU the multiplication or division operation is carried out in a specific way and is determined by the write access sequence of the six MDU data registers, MDUWR0~MDUWR5. The low byte data, regardless of the dividend, multiplicand, divisor or multiplier, must first be written into the corresponding MDU data register followed by the high byte data. All MDU operations will be executed after the MDUWR5 register is write-accessed together with the correct specific write access sequence of the MDUWRn. Note that it is not necessary to consecutively write data into the MDU data registers but must be in a correct write access sequence. Therefore, a non-write MDUWRn instruction or an interrupt, etc., can be inserted into the correct write access sequence without destroying the write operation. The relationship between the write access sequence and the MDU operation is shown in the following.

- 32-bit/16-bit Division Operation: Write data sequentially into the six MDU data registers from MDUWR0 to MDUWR5.
- 16-bit/16-bit Division Operation: Write data sequentially into the specific four MDU data registers in a sequence of MDUWR0, MDUWR1, MDUWR4 and MDUWR5 with no write access to MDUWR2 and MDUWR3.
- 16-bit/16-bit Multiplication Operation: Write data sequentially into the specific four MDU data register in a sequence of MDUWR0, MDUWR4, MDUWR1 and MDUWR5 with no write access to MDUWR2 and MDUWR3.

After the specific write access sequence is determined, the MDU will start to perform the corresponding operation. The calculation time necessary for these MDU operations are different. During the calculation time any read/write access to the six MDU data registers is forbidden. After the completion of each operation, it is necessary to check the operation status in the MDUWCTRL register to make sure that whether the operation is correct or not. Then the operation result can be read out from the corresponding MDU data registers in a specific read access sequence if the operation is correctly finished. The necessary calculation time for different MDU operations is listed in the following.

- 32-bit/16-bit division operation: 17 × t_{SYS}.
- 16-bit/16-bit division operation: 9 × t_{SYS}.
- 16-bit/16-bit multiplication operation: 11 × t_{SYS}.

The operation results will be stored in the corresponding MDU data registers and should be read out from the MDU data registers in a specific read access sequence after the operation is completed. Noe that it is not necessary to consecutively read data out from the MDU data registers but must be in a correct read access sequence. Therefore, a non-read MDUWRn instruction or an interrupt, etc., can be inserted into the correct read access sequence without destroying the read operation. The relationship between the operation result read access sequence and the MDU operation is shown in the following.

- 32-bit/16-bit division operation: Read the quotient from MDUWR0 to MDUWR3 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit/16-bit division operation: Read the quotient from MDUWR0 and MDUWR1 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit/16-bit multiplication operation: Read the product sequentially from MDUWR0 to MDUWR3.

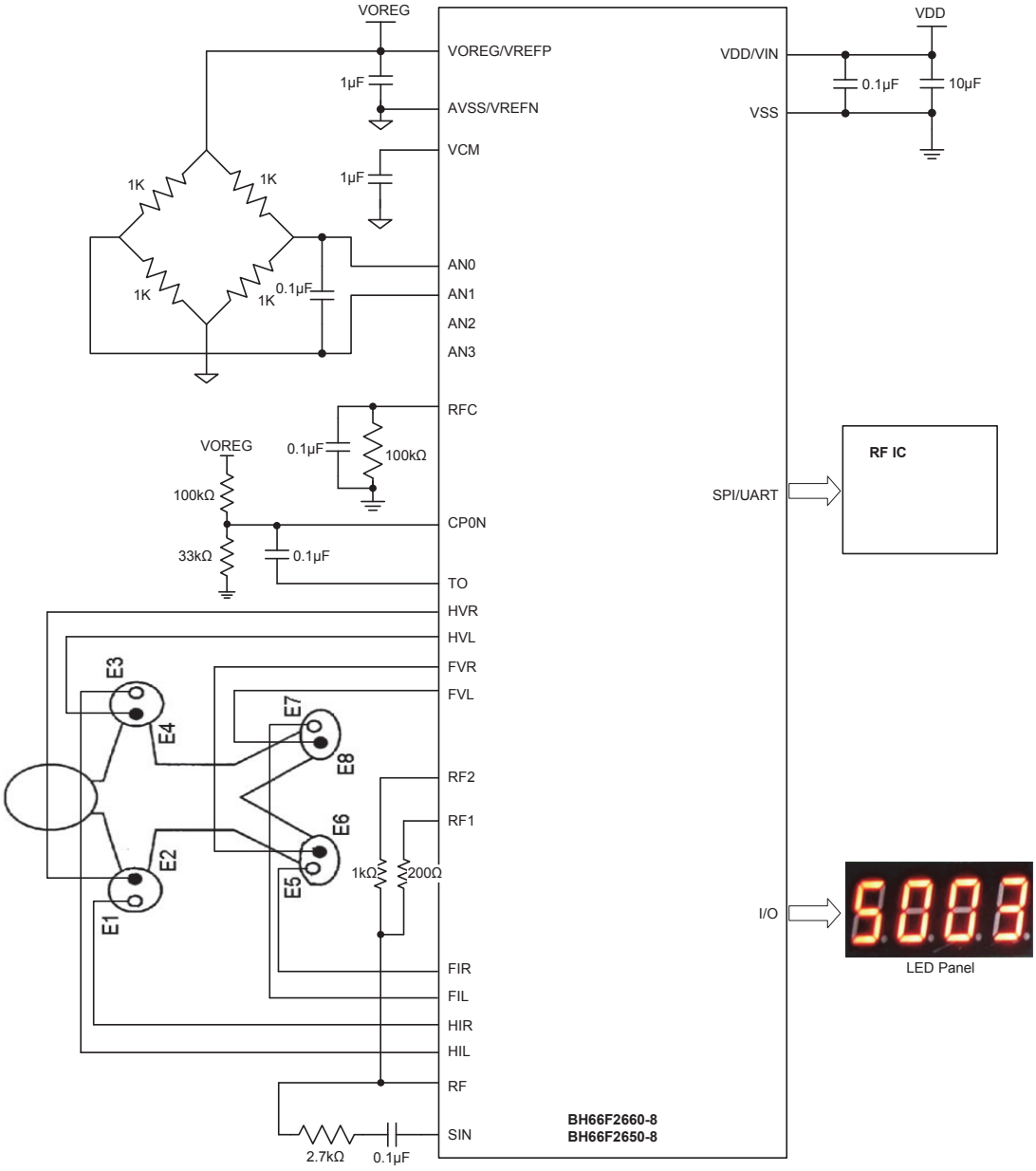
The overall important points for the MDU read/write access sequence and calculation time are summarized in the following table.

Operations / Items	32-bit / 16-bit Division	16-bit / 16-bit Division	16-bit × 16-bit Multiplication
Write Sequence First write ↓ ↓ ↓ ↓ Last write	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Dividend Byte 2 written to MDUWR2 Dividend Byte 3 written to MDUWR3 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Multiplicand Byte 0 written to MDUWR0 Multiplier Byte 0 written to MDUWR4 Multiplicand Byte 1 written to MDUWR1 Multiplier Byte 1 written to MDUWR5
Calculation Time	17 × t_{SYS}	9 × t_{SYS}	11 × t_{SYS}
Read Sequence First read ↓ ↓ ↓ ↓ Last read	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Quotient Byte 2 read from MDUWR2 Quotient Byte 3 read from MDUWR3 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Product Byte 0 written to MDUWR0 Product Byte 1 written to MDUWR1 Product Byte 2 written to MDUWR2 Product Byte 3 written to MDUWR3

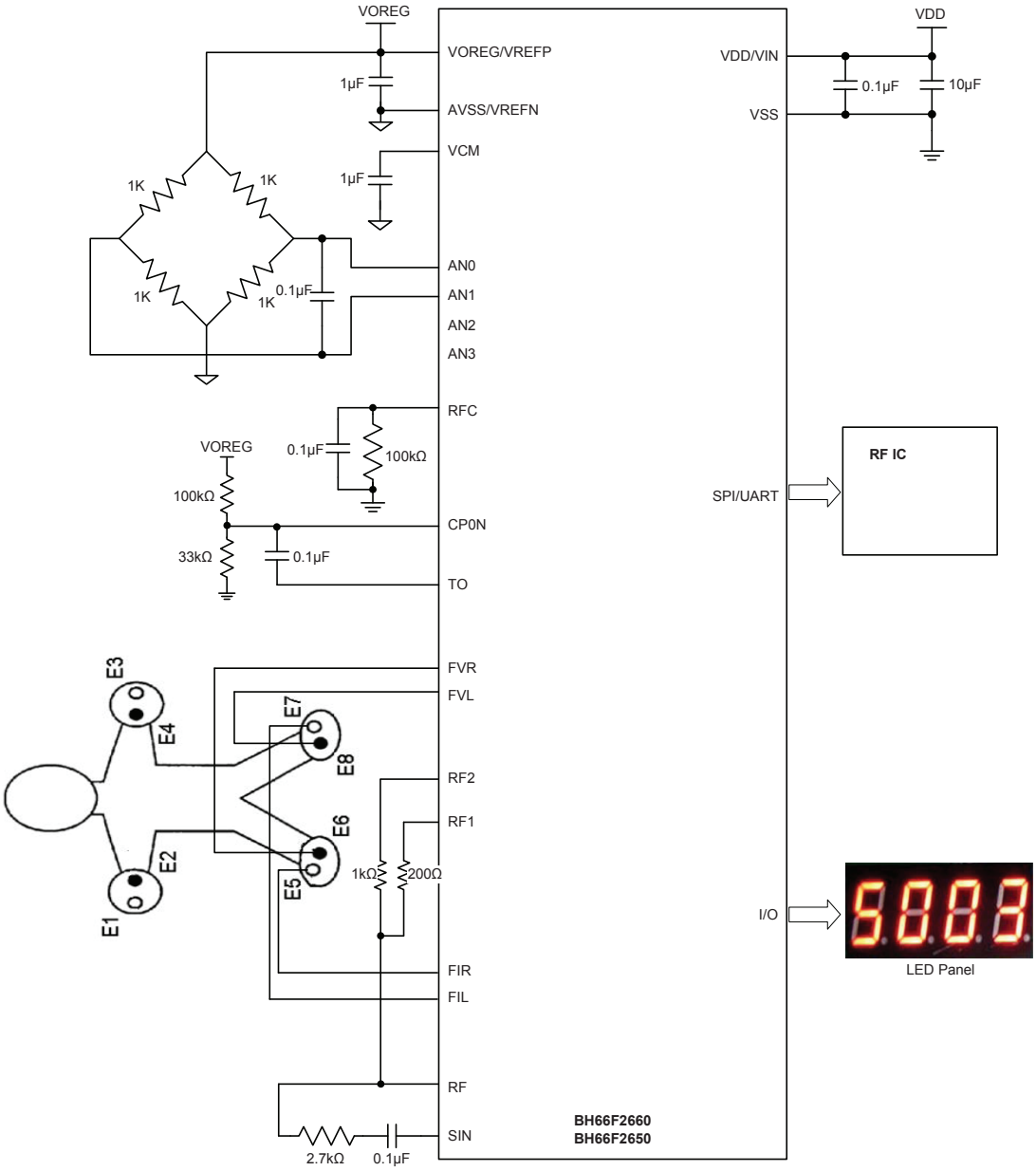
MDU Operations Summary

Application Circuits

8-electrode



4-electrode


VOREG
VOREG/VREFP
1μF
AVSS/VREFN
VCM
1μF
1K
1K
1K
1K
0.1μF
AN0
AN1
AN2
AN3
RFC
VOREG
0.1μF
100kΩ
100kΩ
CP0N
33kΩ
0.1μF
TO
FVR
FVL
E3
E4
E7
E8
RF2
RF1
E6
E2
E5
E1
1kΩ
200Ω
FIR
FIL
RF
SIN
2.7kΩ
0.1μF
BH66F2660
BH66F2650
VDD
VDD/VIN
0.1μF
10μF
VSS
RF IC
SPI/UART
I/O
LED Panel

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 0.5μs and branch or call instructions would be implemented within 1μs. 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 instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV, SC
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV, SC
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV, SC
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV, SC
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV, SC
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV, SC, CZ
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV, SC, CZ
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV, SC, CZ
SBC A,x	Subtract immediate data from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	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	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch Operation			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m]	Skip if Data Memory is not zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read Operation			
TABRD [m]	Read table (specific page) to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
ITABRD [m]	Increment table pointer TBLP first and Read table to TBLH and Data Memory	2Note	None
ITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	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 three 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 WDT" instruction the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after the "CLR WDT" instructions is executed. Otherwise the TO and PDF flags remain unchanged.

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sections except sector 0, the extended instruction can be used to access the data memory instead of using the indirect addressing access to improve the CPU firmware performance.

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	2Note	Z, C, AC, OV, SC
LADC A,[m]	Add Data Memory to ACC with Carry	2	Z, C, AC, OV, SC
LADCM A,[m]	Add ACC to Data memory with Carry	2Note	Z, C, AC, OV, SC
LSUB A,[m]	Subtract Data Memory from ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LSBC A,[m]	Subtract Data Memory from ACC with Carry	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2Note	C
Logic Operation			
LAND A,[m]	Logical AND Data Memory to ACC	2	Z
LOR A,[m]	Logical OR Data Memory to ACC	2	Z
LXOR A,[m]	Logical XOR Data Memory to ACC	2	Z
LANDM A,[m]	Logical AND ACC to Data Memory	2Note	Z
LORM A,[m]	Logical OR ACC to Data Memory	2Note	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2Note	Z
LCPL [m]	Complement Data Memory	2Note	Z
LCPLA [m]	Complement Data Memory with result in ACC	2	Z
Increment & Decrement			
LINCA [m]	Increment Data Memory with result in ACC	2	Z
LINC [m]	Increment Data Memory	2Note	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2Note	Z
Rotate			
LRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2Note	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2Note	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2Note	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2Note	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2Note	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2Note	None
LSET [m].i	Set bit of Data Memory	2Note	None

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

Note: 1. For these extended skip instructions, if the result of the comparison involves a skip then up to four cycles are required, if no skip takes place two cycles is required.

2. Any extended instruction which changes the contents of the PCL register will also require three cycles for execution.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

ADCM A,[m] Add ACC to Data Memory with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

ADD A,[m] Add Data Memory to ACC

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

ADD A,x Add immediate data to ACC

Description The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation ACC ← ACC + x

Affected flag(s) OV, Z, AC, C, SC

ADDM A,[m] Add ACC to Data Memory

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

AND A,[m] Logical AND Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC "AND" x

Affected flag(s) Z

ANDM A,[m] Logical AND ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

Description Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

DEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

Description This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

OR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" x

Affected flag(s) Z

ORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

RET Return from subroutine

Description The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation Program Counter ← Stack

Affected flag(s) None

RET A,x Return from subroutine and load immediate data to ACC

Description The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

Description The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

RLA [m] Rotate Data Memory left with result in ACC

Description: The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s): None

RLC [m] Rotate Data Memory left through Carry

Description: The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation:
[m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s): C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s): C

RR [m] Rotate Data Memory right

Description: The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation:
[m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s): None

RRA [m] Rotate Data Memory right with result in ACC

Description: Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

Description: The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation:
[m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s): C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s): C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: $ACC \leftarrow ACC - [m] - \overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

SBC A, x Subtract immediate data from ACC with Carry

Description: The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: $ACC \leftarrow ACC - [m] - \overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: $[m] \leftarrow ACC - [m] - \overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

SDZ [m] Skip if decrement Data Memory is 0

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation:
[m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation:
ACC ← [m] − 1
Skip if ACC=0

Affected flag(s): None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

SNZ [m].i Skip if Data Memory is not 0

Description If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

Description If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m]≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − x

Affected flag(s) OV, Z, AC, C, SC, CZ

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

SZ [m] 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

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

TABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer pair (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRD [m] Increment table pointer low byte first and read table to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

XOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Extended Instruction Definition

The extended instructions are used to directly access the data stored in any data memory sections.

LADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

LADCM A,[m] Add ACC to Data Memory with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

LADD A,[m] Add Data Memory to ACC

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

LADDM A,[m] Add ACC to Data Memory

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

LAND A,[m] Logical AND Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

LANDM A,[m] Logical AND ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″AND″ [m]

Affected flag(s) Z

LCLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

LCLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

LCPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation $[m] \leftarrow \overline{[m]}$

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow \overline{[m]}$

Affected flag(s) Z

LDAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

LDEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation $[m] \leftarrow [m] - 1$

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

LINC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation $[m] \leftarrow [m] + 1$

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] + 1$

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

LMOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

LOR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

LORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

LRL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

LRLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

LRLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

LRLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

LRR [m] Rotate Data Memory right

Description: The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation:
[m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s): None

LRRA [m] Rotate Data Memory right with result in ACC

Description: Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s): None

LRRC [m] Rotate Data Memory right through Carry

Description: The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation:
[m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s): C

LRRCA [m] Rotate Data Memory right through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s): C

LSBC A,[m] Subtract Data Memory from ACC with Carry

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: $ACC \leftarrow ACC - [m] - \overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

LSBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: $[m] \leftarrow ACC - [m] - \overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

LSDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

LSET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

LSET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

LSIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

LSIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

LSNZ [m].i Skip if Data Memory is not 0

Description If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

Description If the content of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m] ≠ 0

Affected flag(s) None

LSUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

LSUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

LSWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

LSWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

LSZ [m] Skip if Data Memory is 0

Description 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

LSZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

LSZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a 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

LTABRD [m] Read table (current page) to TBLH and Data Memory

Description The low byte of the program code (current 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

LTABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRD [m] Increment table pointer low byte first and read table to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LXOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

LXORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

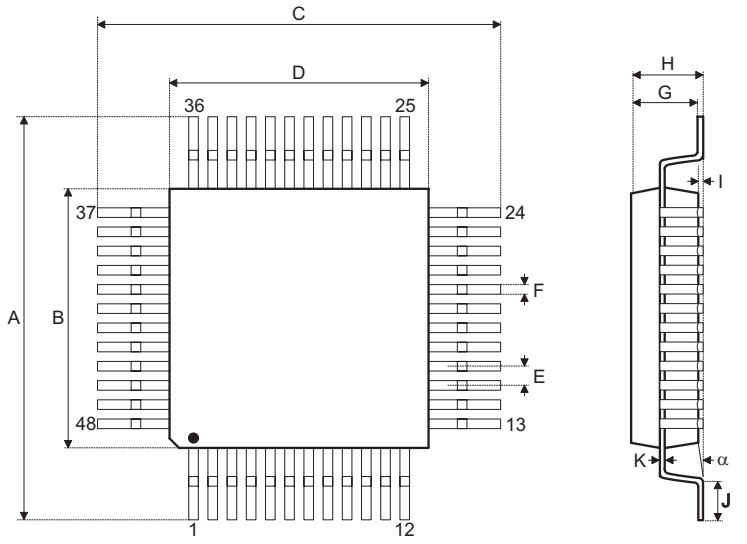
Package Information

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

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

- Further Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- Packing Meterials Information
- Carton information

48-pin LQFP (7mm×7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.020 BSC	—
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.50 BSC	—
F	0.17	0.22	0.27
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°