Enhanced Smoke Detector Flash MCU with Buzzer Driver

BA45F25363

Revision: V1.00 Date: February 26, 2025

www.holtek.com

Features

CPU Features

- Operating Voltage
  - f_{SYS}=2MHz: 2.2V~5.5V
  - 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
- Oscillator types
  - Internal High Speed 2/4/8MHz RC – HIRC
  - Internal Low Speed 32.768kHz RC – LIRC
  - External High Speed Crystal – HXT
  - External Low Speed 32.768kHz Crystal – LXT
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 12-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 16K×16
- RAM Data Memory: 2048×8
- True EEPROM Memory: 256×8
- Watchdog Timer function
- In Application Programming – IAP
- 31 bidirectional I/O lines
- Two external interrupt lines shared with I/O pins
- Programmable I/O port source current for LED applications
- Sink current generator for constant current output
- Smoke Detector AFE including two operational amplifiers
- Power Line Transceiver including two comparators, one Operational Amplifier and three D/A converters
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output or single pulse output function
- Dual Time Base functions for generation of fixed time interrupt signals
- Serial Interface Module – SIM for SPI or I^2C communication

- Dual Fully-duplex/Half-duplex Universal Asynchronous Receiver and Transmitter Interfaces – UART
- 16 external channel 12-bit resolution A/D converter with Internal Reference Voltage V_{BGREF}
- Temperature Sensor with internal reference voltage
- 16-bit Voice D/A converter
- Integrated 16-bit Cyclic Redundancy Check function – CRC
- Low Voltage Reset function
- Low Voltage Detect function
- Package type: 48-pin LQFP

Buzzer Driver Features

- Input voltage (BDV_{DD}): 2.2V~5.5V
- Integrated 2-pin and 3-pin buzzer drivers
- Integrated a boost converter
- Integrated a 5V regulator
- Buzzer watchdog timer (BWDT) for MCU failure alarm function

General Description

The BA45F25363 is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller, designed for smoke detector applications.

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

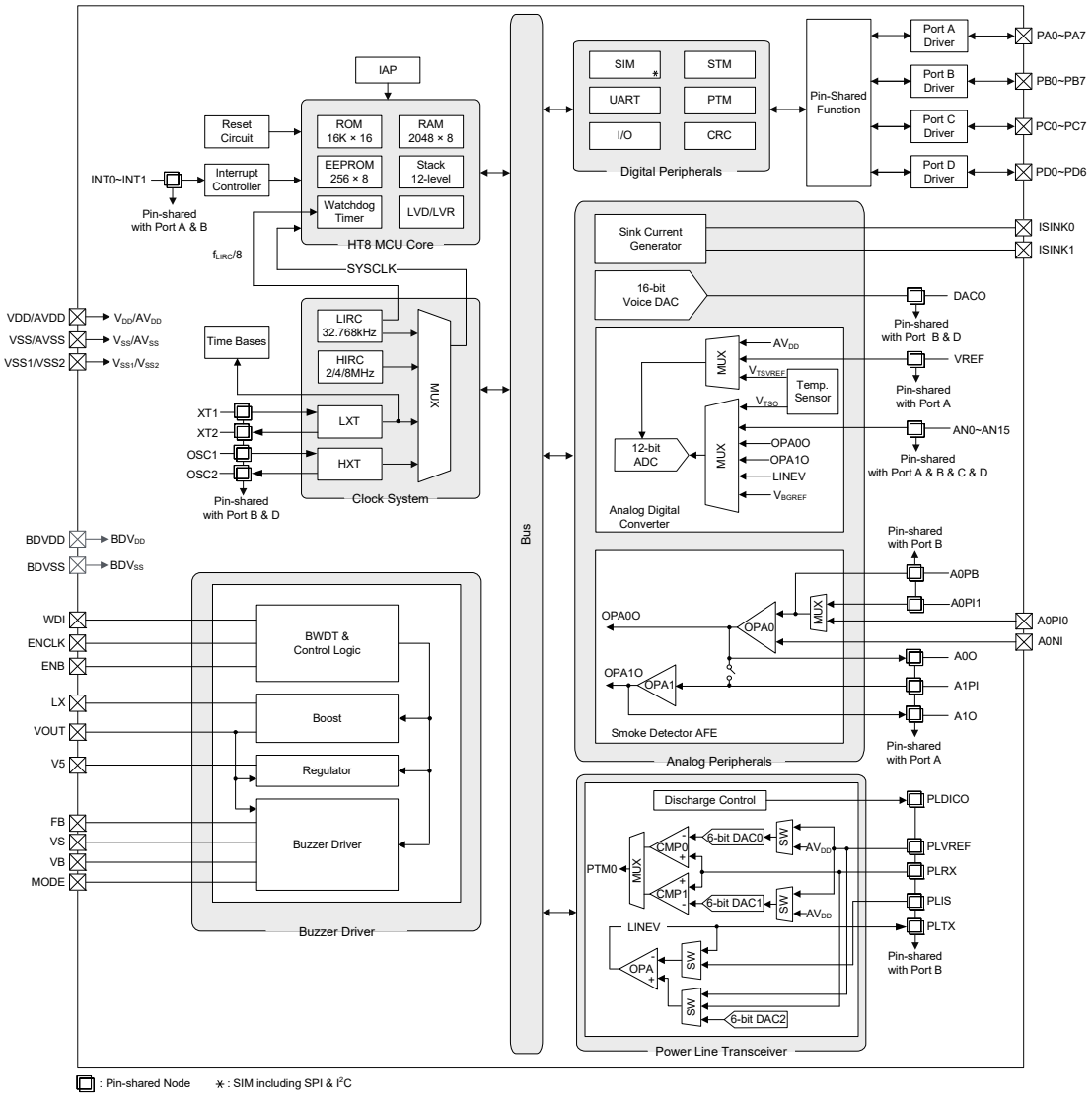
Analog features include a multi-channel Analog to Digital converter with an integrated temperature sensor circuitry, a Smoke Detector AFE, a Sink Current generator and a Power Line Transceiver including two comparators, one Operational Amplifier and three D/A converters. With regard to internal timers, the device includes multiple and extremely flexible Timer Modules providing functions for timing, pulse generation and PWM output operations. Communication with the outside world is catered for by including a fully integrated SPI, I^2C and UART interface functions, three popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

The device also includes a buzzer driver with a boost converter, a 5V regulator and a buzzer watchdog. The buzzer driver supports 2-pin or 3-pin buzzers, the 5V regulator is used to drive the blue LED, and the buzzer watchdog provides the MCU failure alarm function.

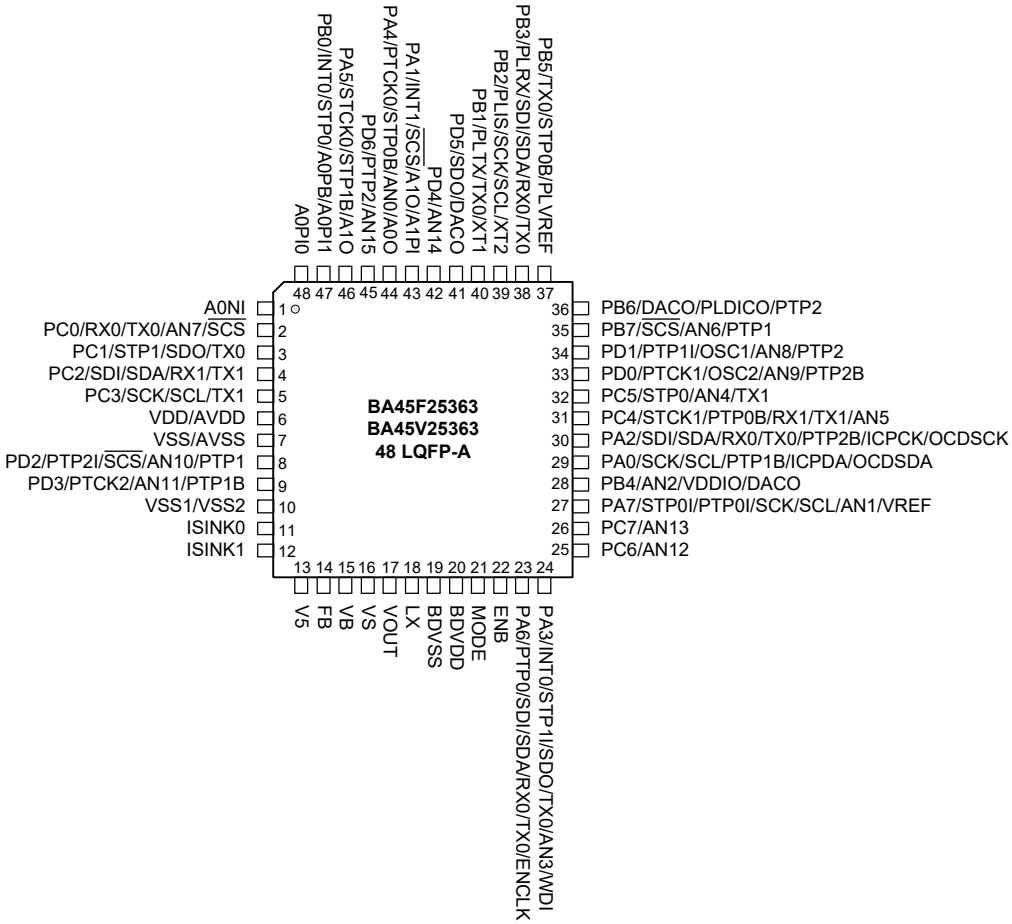
A full choice of external, internal high and low speed oscillators is provided including two fully integrated system oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources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

This inclusion of flexible I/O programming features, 16-bit Cyclic Redundancy Check function, 16-bit Voice D/A converter and Time Base functions along with many other features ensure that the device will find excellent use in the smoke detector applications.

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 OCDS dedicated pins and as such only available for the BA45V25363 device which is the OCDS EV chip for the BA45F25363 device.

3. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input conditions. Refer to the "Standby Current Considerations" and "Input/Output Ports" sections.

Pin Description

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

Pin Name	Function	OPT	I/T	O/T	Description
PA0/SCK/SCL/PTP1B/ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	SCK	PAS0 IFS0	ST	CMOS	SPI serial clock
	SCL	PAS0 IFS0	ST	NMOS	I^2C clock line
	PTP1B	PAS0	—	CMOS	PTM1 inverting output
	ICPDA	—	ST	CMOS	ICP data/address
	OCDSDA	—	ST	CMOS	OCDS data/address, for EV chip only
PA1/INT1/$\overline{SCS}$/A1O/A1PI	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT1	PAS0 INTC0 INTEG	ST	—	External interrupt input 1
	$\overline{SCS}$	PAS0 IFS0	ST	CMOS	SPI slave select
	A1O	PAS0	—	AN	SD OPA1 output
	A1PI	PAS0	AN	—	SD OPA1 positive input
PA2/SDI/SDA/RX0/TX0/PTP2B/ICPCK/OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	SDI	PAS0 IFS0	ST	—	SPI serial data input
	SDA	PAS0 IFS0	ST	NMOS	I^2C data line
	RX0/TX0	PAS0 IFS1	ST	CMOS	UART0 serial data input in full-duplex communication or UART0 serial data input/output in Single Wire Mode communication
	PTP2B	PAS0	—	CMOS	PTM2 inverting output
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/INT0/STP1I/SDO/TX0/AN3/WDI	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT0	PAS0 IFS1 INTC0 INTEG	ST	—	External Interrupt 0
	STP1I	PAS0	ST	—	STM1 capture input
	SDO	PAS0	—	CMOS	SPI serial data output
	TX0	PAS0	—	CMOS	UART0 serial data output
	AN3	PAS0	AN	—	A/D Converter external input channel 3
	WDI	—	ST	—	BWDT Input With Pull-down Resistor

Pin Name	Function	OPT	I/T	O/T	Description
PA4/PTCK0/STP0B/AN0/A0O	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	PTCK0	PAS1	ST	—	PTM0 clock input or capture input
	STP0B	PAS1	—	CMOS	STM0 inverting output
	AN0	PAS1	AN	—	A/D Converter external input channel 0
	A0O	PAS1	—	AN	SD OPA0 output
PA5/STCK0/STP1B/A1O	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	STCK0	PAS1	ST	—	STM0 clock input
	STP1B	PAS1	—	CMOS	STM1 inverting output
	A1O	PAS1	—	AN	SD OPA1 output
PA6/PTP0/SDI/SDA/RX0/TX0/ENCLK	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	PTP0	PAS1	—	CMOS	PTM0 output
	SDI	PAS1 IFS0	ST	—	SPI serial data input
	SDA	PAS1 IFS0	ST	NMOS	I^2C data line
	RX0/TX0	PAS1 IFS1	ST	CMOS	UART0 serial data input in full-duplex communication or UART0 serial data input/output in Single Wire Mode communication
	ENCLK	—	ST	—	ENCLK Control Boost/Regulator/Buzzer
PA7/STP0I/PTP0I/SCK/SCL/AN1/VREF	PA7	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	STP0I	PAS1	ST	—	STM0 capture input
	PTP0I	PAS1 IFS0	ST	—	PTM0 capture Input
	SCK	PAS1 IFS0	ST	CMOS	SPI serial clock
	SCL	PAS1 IFS0	ST	NMOS	I^2C clock line
	AN1	PAS1	AN	—	A/D Converter external input channel 1
	VREF	PAS1	AN	—	A/D Converter external reference voltage
PB0/INT0/STP0/A0PB/A0PI1	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	INT0	PBS0 IFS1 INTC0 INTEG	ST	—	External interrupt input 0
	STP0	PBS0	—	CMOS	STM0 output
	A0PB	PBS0	AN	—	SD OPA0 bias input
	A0PI1	PBS0	AN	—	SD OPA0 positive input channel 1
PB1/PLTX/TX0/XT1	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PLTX	PBS0	—	AN	Power Line Transceiver TX output
	TX0	PBS0	—	CMOS	UART0 serial data output
	XT1	PBS0	AN	—	LXT oscillator pin

Pin Name	Function	OPT	I/T	O/T	Description
PB2/PLIS/SCK/SCL/XT2	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PLIS	PBS0	AN	—	Power Line Transceiver IS input
	SCK	PBS0 IFS0	ST	CMOS	SPI serial clock
	SCL	PBS0 IFS0	ST	NMOS	I^2C clock line
	XT2	PBS0	—	AN	LXT oscillator pin
PB3/PLRX/SDI/SDA/RX0/TX0	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PLRX	PBS0	AN	—	Power Line Transceiver RX input
	SDI	PBS0 IFS0	ST	—	SPI serial data input
	SDA	PBS0 IFS0	ST	NMOS	I^2C data line
	RX0/TX0	PBS0 IFS1	ST	CMOS	UART0 serial data input in full-duplex communication or UART0 serial data input/output in Single Wire Mode communication
PB4/AN2/VDDIO/DACO	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	AN2	PBS1	AN	—	A/D Converter external input channel 2
	VDDIO	PBS1	PWR	—	Power for PA0, PA2, PA3, PA6, PA7, PC4, PC5
	DACO	PBS1	—	AN	16-bit D/A Converter output
PB5/TX0/STP0B/PLVREF	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	TX0	PBS1	—	CMOS	UART0 serial data output
	STP0B	PBS1	—	CMOS	STM0 inverting output
	PLVREF	PBS1	AN	—	Power Line Transceiver D/A Converter reference
PB6/DACO/PLDICO/PTP2	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	DACO	PBS1	—	AN	16-bit D/A Converter output
	PLDICO	PBS1	—	AN	Power Line Transceiver discharge control
	PTP2	PBS1	—	CMOS	PTM2 output
PB7/$\overline{SCS}$/AN6/PTP1	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	$\overline{SCS}$	PBS1 IFS0	ST	CMOS	SPI slave device select
	AN6	PBS1	AN	—	A/D Converter external input channel 6
	PTP1	PBS1	—	CMOS	PTM1 output
PC0/RX0/TX0/AN7/$\overline{SCS}$	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	RX0/TX0	PCS0 IFS1	ST	CMOS	UART0 serial data input in full-duplex communication or UART0 serial data input/output in Single Wire Mode communication
	AN7	PCS0	AN	—	A/D Converter external input channel 7
	$\overline{SCS}$	PCS0 IFS0	ST	CMOS	SPI slave device select
PC1/STP1/TX0/SDO	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	STP1	PCS0	—	CMOS	STM1 output
	TX0	PCS0	—	CMOS	UART0 serial data output
	SDO	PCS0	—	CMOS	SPI serial data output

Pin Name	Function	OPT	I/T	O/T	Description
PC2/SDI/SDA/RX1/TX1	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SDI	PCS0 IFS0	ST	—	SPI serial data input
	SDA	PCS0 IFS0	ST	NMOS	I^2C data line
	RX1/TX1	PCS0 IFS1	ST	CMOS	UART1 serial data input in full-duplex communication or UART1 serial data input/output in Single Wire Mode communication
PC3/SCK/SCL/TX1	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SCK	PCS0 IFS0	ST	CMOS	SPI serial clock
	SCL	PCS0 IFS0	ST	NMOS	I^2C clock line
	TX1	PCS0	—	CMOS	UART1 serial data output
PC4/STCK1/PTP0B/RX1/TX1/AN5	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	STCK1	PCS1	ST	—	STM1 clock input
	PTP0B	PCS1	—	CMOS	PTM0 inverting output
	RX1/TX1	PCS1 IFS1	ST	CMOS	UART1 serial data input in full-duplex communication or UART1 serial data input/output in Single Wire Mode communication
	AN5	PCS1	AN	—	A/D Converter external input channel 5
PC5/STP0/AN4/TX1	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	STP0	PCS1	—	CMOS	STM0 output
	AN4	PCS1	AN	—	A/D Converter external input channel 4
	TX1	PCS1	—	CMOS	UART1 serial data output
PC6/AN12	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	AN12	PCS1	AN	—	A/D Converter external input channel 12
PC7/AN13	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	AN13	PCS1	AN	—	A/D Converter external input channel 13
PD0/PTCK1/OSC2/AN9/PTP2B	PD0	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTCK1	PDS0	ST	—	PTM1 clock input or capture input
	OSC2	PDS0	—	AN	HXT oscillator pin
	AN9	PDS0	AN	—	A/D Converter external input channel 9
	PTP2B	PDS0	—	CMOS	PTM2 inverting output
PD1/PTP1I/OSC1/AN8/PTP2	PD1	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP1I	PDS0	ST	—	PTM1 capture input
	OSC1	PDS0	AN	—	HXT oscillator pin
	AN8	PDS0	AN	—	A/D Converter external input channel 8
	PTP2	PDS0	—	CMOS	PTM2 output

Pin Name	Function	OPT	I/T	O/T	Description
PD2/PTP2I/$\overline{SCS}$/AN10/PTP1	PD2	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP2I	PDS0	ST	—	PTM2 capture input
	$\overline{SCS}$	PDS0 IFS0	ST	CMOS	SPI slave device select
	AN10	PDS0	AN	—	A/D Converter external input channel 10
	PTP1	PDS0	—	CMOS	PTM1 output
PD3/PTCK2/AN11/PTP1B	PD3	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTCK2	PDS0	ST	—	PTM2 clock input or capture input
	AN11	PDS0	AN	—	A/D Converter external input channel 11
	PTP1B	PDS0	—	CMOS	PTM1 inverting output
PD4/AN14	PD4	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	AN14	PDS1	AN	—	A/D Converter external input channel 14
PD5/SDO/DACO	PD5	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SDO	PDS1	—	CMOS	SPI serial data output
	DACO	PDS1	—	AN	16-bit D/A Converter output
PD6/PTP2/AN15	PD6	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP2	PDS1	—	CMOS	PTM2 output
	AN15	PDS1	AN	—	A/D Converter external input channel 15
ISINK0	ISINK0	—	—	AN	Sink current 0 source
ISINK1	ISINK1	—	—	AN	Sink current 1 source
A0NI	A0NI	—	AN	—	SD OPA0 negative input
A0PI0	A0PI0	—	AN	—	SD OPA0 positive input channel 0
VDD/AVDD	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
VSS/AVSS	VSS	—	PWR	—	Digital negative power supply
	AVSS	—	PWR	—	Analog negative power supply
VSS1/VSS2	VSS1	—	PWR	—	Sink Current Generator negative power supply
	VSS2	—	PWR	—	Sink Current Generator negative power supply
BDVDD	BDVDD	—	PWR	—	Buzzer positive power supply
BDVSS	BDVSS	—	PWR	—	Buzzer negative power supply, ground
ENB	ENB	—	ST	—	Enable pin, control Boost/Regulator/Buzzer
MODE	MODE	—	ST	—	MODE pin, control Boost/Regulator/Buzzer
VOUT	VOUT	—	—	PWR	Boost converter output
LX	LX	—	—	PWR	Power wwitch output
V5	V5	—	—	PWR	Regulator output
FB	FB	—	ST	—	Feedback pin in 3-pin buzzer mode
VS	VS	—	—	CMOS	Output for buzzer
VB	VB	—	—	CMOS	Complementary output for buzzer

Legend: I/T: Input type;
OPT: Optional by register option;
ST: Schmitt Trigger input;
NMOS: NMOS output;
O/T: Output type;
PWR: Power;
CMOS: CMOS output;
AN: Analog signal.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}-0.3V to 6.0V
Input Voltage ..... V_{SS}-0.3V to V_{DD}+0.3V
Storage Temperature..... -60°C to 150°C
Operating Temperature..... -40°C to 85°C
I_{OL} Total 80mA
I_{OH} Total ..... -80mA
VOUT, FB, VB, VS..... V_{SS}-0.3V to 18V
Total Power Dissipation 500mW

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

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	f_{SYS}=2MHz	2.2	—	5.5	V
		f_{SYS}=4MHz	2.2	—	5.5	
		f_{SYS}=8MHz	2.2	—	5.5	
	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 – LXT	f_{SYS}=32.768kHz	2.2	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32.768kHz	2.2	—	5.5	V

Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode (LIRC)	2.2V	f_{SYS}=32.768kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	SLOW Mode (LXT)	2.2V	f_{SYS}=32.768kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode (HIRC)	2.2V	f_{SYS}=2MHz	—	0.15	0.20	mA
		3V		—	0.2	0.3	
		5V		—	0.4	0.6	
		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	
	FAST Mode (HXT)	2.2V	f_{SYS}=4MHz	—	0.4	0.6	mA
		3V		—	0.50	0.75	
		5V		—	1.0	1.5	
		2.2V	f_{SYS}=8MHz	—	0.8	1.2	mA
		3V		—	1.0	1.5	
		5V		—	2	3	
		2.7V	f_{SYS}=12MHz	—	1.2	2.2	mA
		3V		—	1.50	2.75	
		5V		—	3.0	4.5	
		3.3V	f_{SYS}=16MHz	—	3.2	4.8	mA
		5V		—	4	6	

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

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

Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3	5	6	
	IDLE0 Mode (LIRC)	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE0 Mode (LXT)	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE1 Mode (HIRC)	2.2V	f_{SUB} on, f_{SYS}=2MHz	—	60	120	140	μA
		3V		—	70	140	160	
		5V		—	130	260	280	
		2.2V	f_{SUB} on, f_{SYS}=4MHz	—	144	200	240	μA
		3V		—	180	250	300	
		5V		—	400	600	720	
		2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	
	IDLE1 Mode (HXT)	2.2V	f_{SUB} on, f_{SYS}=4MHz	—	144	200	240	μA
		3V		—	180	250	300	
		5V		—	400	600	720	
		2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	432	720	765	μA
		3V		—	540	900	955	
		5V		—	800	1440	1525	
		3.3V	f_{SUB} on, f_{SYS}=16MHz	—	1.1	1.6	1.9	mA
		5V		—	1.4	2.0	2.4	

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

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

A.C. Characteristics

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

Internal High Speed Oscillator – HIRC – Frequency Accuracy

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	2MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	2	+1%	MHz
			-10°C~50°C	-2%	2	+2%	
			-20°C~60°C	-3%	2	+3%	
			-40°C~85°C	-4%	2	+4%	
		2.2V~5.5V	25°C	-6%	2	+9%	
			-40°C~85°C	-6%	2	+10%	
	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%	

Note: 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%.

Internal Low Speed Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	3V	-10°C~50°C	-2%	32.768	+2%	kHz
			-40°C~85°C	-4%	32.768	+4%	
		2.2V~5.5V	-40°C~85°C	-10%	32.768	+10%	
t_{START}	LIRC Start-up Time	—	-40°C~85°C	—	—	100	μs

External Low Speed Crystal Oscillator Characteristics – LXT

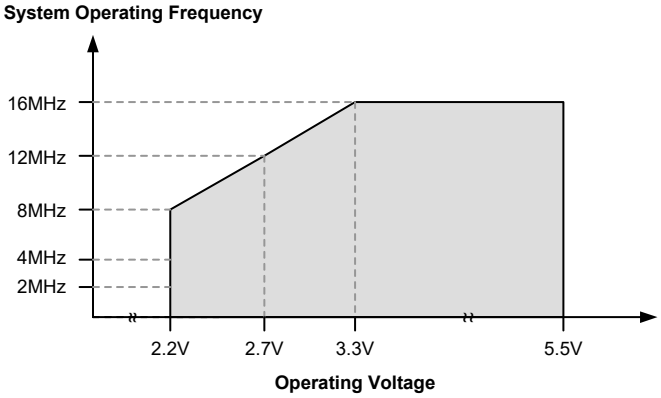
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{LXT}	Oscillator Frequency	2.2V~5.5V	—	—	32768	—	Hz
t_{START}	LXT Start-up Time	3V	—	—	—	1000	ms
		5V	—	—	—	1000	ms

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Duty Cycle	Duty Cycle	—	—	40	—	60	%
R_{NEG}	Negative Resistance(Note)	2.2V	—	3×ESR	—	—	Ω

Note: C1, C2 and R_P are external components. C1=C2=10pF. R_P=10MΩ, C_L=7pF, ESR=30kΩ.

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{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_{SYS}
		—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LXT}$	—	1024	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	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_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$ or f_{LXT}	—	2	—	t_{SYS}
	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	14	16	18	ms
	System Reset Delay Time (WDTC Register Software Reset)	—	—				
	System Reset Delay Time (WDT Overflow Reset)	—	—				
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

Note: 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_{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

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

For PA1, PA4~PA5, PB0~PB7, PC0~PC3, PC6~PC7 and PD0~PD6 pins.

Ta=-40°C~85°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, 2, 4, 6)	-0.7	-1.5	—	mA
		5V		-1.5	-2.9	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=01B (n=0, 1; m=0, 2, 4, 6)	-1.3	-2.5	—	
		5V		-2.5	-5.1	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=10B (n=0, 1; m=0, 2, 4, 6)	-1.8	-3.6	—	
		5V		-3.6	-7.3	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=11B (n=0, 1; m=0, 2, 4, 6)	-4	-8	—	
		5V		-8	-16	—	
R_{PH}	Pull-high Resistance for I/O Ports (1)	3V	—	20	60	90	kΩ
		5V		10	30	45	
I_{LEAK}	Input Leakage Current	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{INT}	Interrupt Pin Minimum Pulse Width	—	—	10	—	—	μs
t_{TCK}	xTCKn Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TPI}	PTP0I Input Pin Minimum Pulse Width	—	—	0.1	—	—	μs
	STPnI/PTP1I/PTP2I Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
f_{TMCLK}	xTMn Maximum Timer Clock Source Frequency	5V	—	—	—	1	f_{SYS}
t_{CPW}	xTMn Minimum Capture Pulse Width	—	—	t_{CPW}(2)	—	—	μs

Note: 1. 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 pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R_{PH} value.

2. For PTMn (n=0~2):

If PTnCAPTS=0, then $t_{CPW}=\max(2\times t_{TMCLK}, t_{TPI})$

If PTnCAPTS=1, then $t_{CPW}=\max(2\times t_{TMCLK}, t_{TCK})$

Ex1: If PTnCAPTS=0, f_{TMCLK}=16MHz

When n=0, t_{TPI}=0.1μs, then $t_{CPW}=\max(0.125\mu s, 0.1\mu s)=0.125\mu s$

When n=1 or 2, t_{TPI}=0.3μs, then $t_{CPW}=\max(0.125\mu s, 0.3\mu s)=0.3\mu s$

Ex2: If PTnCAPTS=1, f_{TMCLK}=16MHz, t_{TCK}=0.3μs, then $t_{CPW}=\max(0.125\mu s, 0.3\mu s)=0.3\mu s$

For STM0 and STM1:

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

Ex1: If f_{TMCLK}=16MHz, t_{TPI}=0.3μs, then $t_{CPW}=\max(0.125\mu s, 0.3\mu s)=0.3\mu s$

Ex2: If f_{TMCLK}=8MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.25μs, 0.3μs)=0.3μs
Ex3: If f_{TMCLK}=4MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.5μs, 0.3μs)=0.5μs
Where $t_{TMCLK}=1/f_{TMCLK}$

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

For PA0, PA2, PA3, PA6, PA7, PC4 and PC5 pins.

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Power Supply – V_{DD0}	—	—	2.2	5.0	5.5	V
V_{DDIO}	Power Supply – V_{DD1}	—	—	2.2	—	V_{DD}	V
V_{IL}	Input Low Voltage for I/O Ports	5V	Pin power=V_{DDn}, V_{DDn}=V_{DD}, n=0~1	0	—	1.5	V
		—	Pin power=V_{DDn}, n=0~1	0	—	0.2V_{DDn}	
V_{IH}	Input High Voltage for I/O Ports	5V	Pin power=V_{DDn}, V_{DDn}=V_{DD}, n=0~1	3.5	—	5.0	V
		—	Pin power=V_{DDn}, n=0~1	0.8V_{DDn}	—	V_{DDn}	
I_{OL}	Sink Current for I/O Ports	3V	V_{OL}=0.1V_{DDn}, V_{DDn}=V_{DD}, n=0~1	16	32	—	mA
		5V	V_{OL}=0.1V_{DDn}, V_{DDn}=V_{DD}, n=0~1	32	65	—	
			V_{OL}=0.1V_{DDn}, V_{DDn}=3V, n=0~1	20	40	—	
I_{OH}	Source Current for I/O Ports	3V	V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=00B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=V_{DD}, n=0~1	-0.7	-1.5	—	mA
		5V	V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=00B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=V_{DD}, n=0~1	-1.5	-2.9	—	
			V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=00B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=3V, n=0~1	-0.40	-0.85	—	
		3V	V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=01B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=V_{DD}, n=0~1	-1.3	-2.5	—	mA
		5V	V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=01B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=V_{DD}, n=0~1	-2.5	-5.1	—	
			V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=01B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=3V, n=0~1	-0.70	-1.35	—	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH}	Source Current for I/O Ports	3V	V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=10B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=V_{DD}, n=0~1	-1.8	-3.6	—	mA
		5V	V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=10B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=V_{DD}, n=0~1	-3.6	-7.3	—	
			V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=10B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=3V, n=0~1	-0.95	-1.90	—	
		3V	V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=11B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=V_{DD}, n=0~1	-4	-8	—	mA
		5V	V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=11B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=V_{DD}, n=0~1	-8	-16	—	
			V_{OH}=0.9V_{DDn}, SLEDCn[m+1:m]=11B (n=0, 1; m=0, 2, 4, 6) V_{DDn}=3V, n=0~1	-2.5	-5.0	—	
R_{PH}	Pull-high Resistance for I/O Ports (1)	3V	V_{DDn}=V_{DD}, n=0~1	20	60	100	kΩ
		5V	V_{DDn}=V_{DD}, n=0~1	10	30	50	
			V_{DDn}=3V, n=0~1	36	110	180	
I_{LEAK}	Input Leakage Current for I/O Ports	5V	V_{IN}=V_{SS} or V_{IN}=V_{DDn}, n=0~1	—	—	±1	μA

Note: 1. 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 pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R_{PH} value.

2. The actual V_{DDn} in the "Conditions" column, which can be V_{DD} or V_{DDIO}, is determined by the "V_{DD}" column value and the individual V_{DDn} voltage range.

3. When VDDn is used as the I/O power, a 0.1μF bypass capacitor should be added close to the VDDn pin.

Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Flash Program Memory							
t_{DEW}	Erase/Write Cycle Time	—	—	—	2	3	ms
I_{DDPGM}	Programming/Erase Current on V_{DD}	—	—	—	—	5	mA
E_P	Cell Endurance	—	—	10K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
Data EEPROM Memory							
t_{EERD}	Read Cycle Time	—	—	—	—	4	t_{SYS}
t_{EEWR}	Write Cycle Time	—	—	—	4	6	ms
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	Data Retention Time	—	Ta=25°C	—	40	—	Year

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
RAM Data Memory							
V_{DR}	Data Retention Voltage	—	—	1.0	—	—	V

Note: "E/W" means Erase/Write times.

LVD/LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable	-5%	2.1	+5%	V
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 2.0V	-5%	2.0	+5%	V
			LVD enable, voltage select 2.2V		2.2		
			LVD enable, voltage select 2.4V		2.4		
			LVD enable, voltage select 2.7V		2.7		
			LVD enable, voltage select 3.0V		3.0		
			LVD enable, voltage select 3.3V		3.3		
			LVD enable, voltage select 3.6V		3.6		
			LVD enable, voltage select 4.0V		4.0		
$I_{LVRLVDBG}$	Operating Current	3V	LVD enable, LVR enable, VBGEN=0	—	—	20	μA
		5V		—	20	25	
		3V	LVD enable, LVR enable, VBGEN=1	—	—	25	μA
		5V		—	25	30	
t_{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	18	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	TLVR[1:0]=00B	120	240	480	μs
		—	TLVR[1:0]=01B	0.5	1.0	2.0	ms
		—	TLVR[1:0]=10B	1	2	4	
		—	TLVR[1:0]=11B	2	4	8	
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	TLVD[1:0]=00B/11B	60	140	220	μs
		—	TLVD[1:0]=01B	90	200	340	
		—	TLVD[1:0]=10B	150	320	580	

Internal Reference Voltage Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{BGREF}	Bandgap Reference Voltage	—	—	-1%	1.2	+1%	V
I_{BGREF}	Operating Current	5.5V	—	—	25	40	μA
PSRR	Power Supply Rejection Ratio	—	Ta=25°C, V_{RIPPLE}=1V_{P-P}, f_{RIPPLE}=100Hz	75	—	—	dB
En	Output Noise	—	Ta=25°C, no load current, f=0.1Hz~10Hz	—	300	—	μV_{RMS}
I_{DRV}	Buffer Driving Capability	—	ΔV_{BGREF}=-1%	1	—	—	mA
I_{SD}	Shutdown Current	—	VBGREN=0	—	—	0.1	μA
t_{START}	Startup Time	2.2V~5.5V	Ta=25°C	—	—	400	μs

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

2. A 0.1μF ceramic capacitor should be connected between VDD and GND.
3. The V_{BGREF} voltage is used as the A/D converter internal signal input.

A/D Converter Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	2.2	—	V_{DD}	V
N_R	Resolution	—	—	—	—	12	Bit
DNL	Differential Non-linearity	—	V_{REF}=V_{DD}, t_{ADCK}=0.5μs, SAMS=0	-3	—	3	LSB
		—	V_{REF}=V_{DD}, t_{ADCK}=4μs, SAMS=1	-3	—	3	LSB
INL	Integral Non-linearity	—	V_{REF}=V_{DD}, t_{ADCK}=0.5μs, SAMS=0	-4	—	4	LSB
		—	V_{REF}=V_{DD}, t_{ADCK}=4μs, SAMS=1	-4	—	4	LSB
I_{ADC}	Additional Current for A/D Converter Enable	2.2V	No load, t_{ADCK}=0.5μs, SAMS=0	—	210	300	μA
		3V		—	240	350	
		5V		—	350	500	
		2.2V	No load, t_{ADCK}=4μs, SAMS=1	—	180	250	μA
		3V		—	200	300	
		5V		—	300	420	
t_{ADCK}	Clock Period	—	AN≠Temperature Sensor, SAMS=0	0.5	—	10.0	μs
		—	AN≠Temperature Sensor, SAMS=1	4	—	10	μs
		2.2V~5.5V	AN=Temperature Sensor, SAMS=0	1	—	2	μs
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t_{ADS}	Sampling Time	—	AN≠Temperature Sensor	—	4	—	t_{ADCK}
		2.2V~5.5V	AN=Temperature Sensor	—	46	—	t_{ADCK}
t_{ADC}	Conversion Time (Including A/D Sample and Hold Time)	—	AN≠Temperature Sensor	—	16	—	t_{ADCK}
		2.2V~5.5V	AN=Temperature Sensor	—	58	—	t_{ADCK}

Temperature Sensor Characteristics

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{TS}	Temperature Sensor Operating Current	3V	TSEN=ADCEN=1, t_{ADCK}=1μs, ADC included, SAMS=0	—	1260	1950	μA
		5V		—	1490	2250	μA
t_{TSS}	Temperature Sensor Turn on Stable Time	3V	—	—	—	100	μs
		5V	—	—	—	100	μs
V_{TSVREF}	Temperature Sensor Reference Voltage	3V	—	-5%	2.01	+5%	V
		5V	—	-5%	2.01	+5%	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
T_{ACC}	Temperature Accuracy (Error)	2.7V~4.5V	V_{REF}=V_{TSVREF}, Ta=0°C~70°C	-2	—	+2	°C
		2.7V~5.5V		-2.5	—	+2.5	
		—		—	±4	—	
		2.7V~4.5V	V_{REF}=V_{TSVREF}, Ta=-40°C~85°C	-4	—	+4	
		—		—	±5	—	

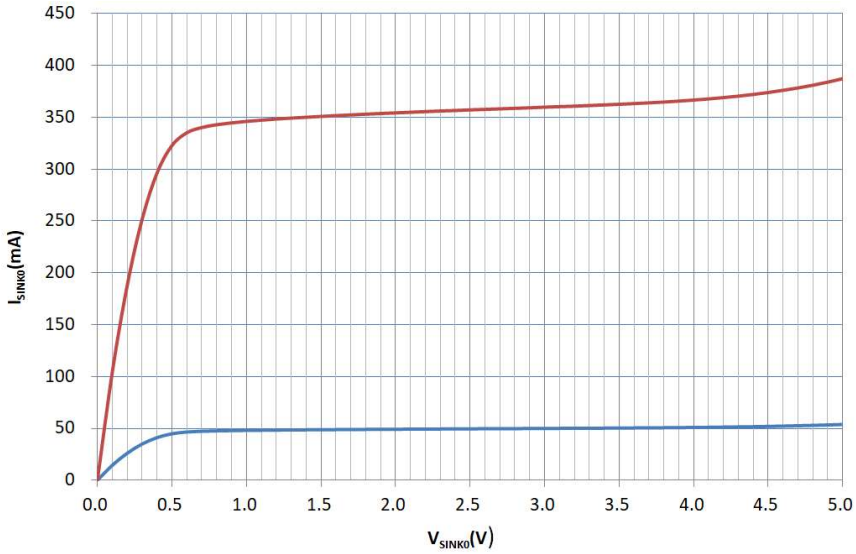
Note: 1. The temperature accuracy T_{ACC} is defined as the error between the actual temperature and the temperature obtained by the conversion of the ADC code through the formula.

2. The relevant characteristics of temperature sensor use normal mode A/D converter (SAMS=0) for voltage conversion.

Sink Current Generator Electrical Characteristics

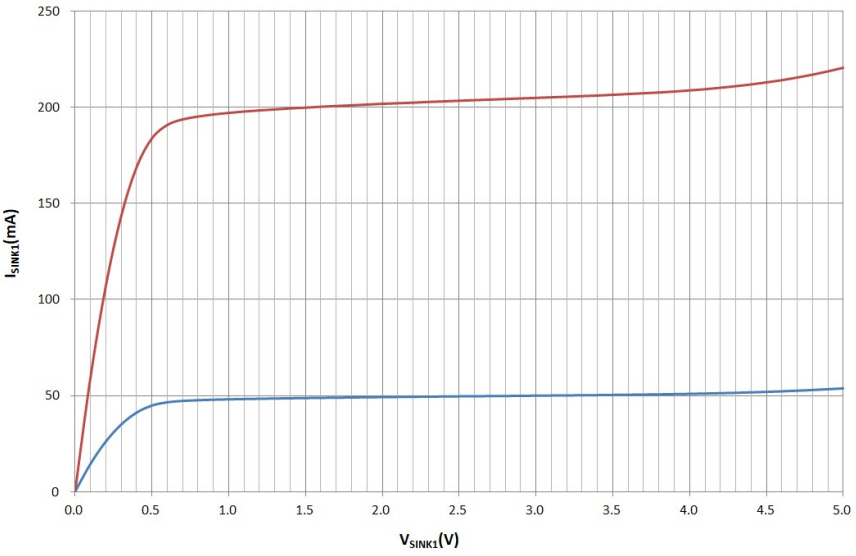
Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{SINK0}	Sink Current for ISINK0 Pin	5V	Ta=25°C, V_{ISINK0}=3.0V, ISGDATA0[4:0]=00000B	47.5	50.0	52.5	mA
		—	Ta=-40°C ~85°C, V_{ISINK0}=1.0V~4.5V, ISGDATA0[4:0]=00000B	41	50	59	
		—	Ta=-40°C ~85°C, V_{ISINK0}=0.7V~1.0V, ISGDATA0[4:0]=00000B	37.5	50.0	50.0	
		5V	Ta=25°C, V_{ISINK0}=3.0V, ISGDATA0[4:0]=11111B	330	360	390	
		—	Ta=-40°C ~85°C, V_{ISINK0}=1.0V~4.5V, ISGDATA0[4:0]=11111B	295	360	425	
		—	Ta=-40°C ~85°C, V_{ISINK0}=0.7V~1.0V, ISGDATA0[4:0]=11111B	270	360	360	
I_{SINK1}	Sink Current for ISINK1 Pin	5V	Ta=25°C, V_{ISINK1}=3.0V, ISGDATA1[4:0]=00000B	43.5	50.0	56.5	mA
		—	Ta=-40°C ~85°C, V_{ISINK1}=1.0V~4.5V, ISGDATA1[4:0]=00000B	41	50	59	
		—	Ta=-40°C ~85°C, V_{ISINK1}=0.7V~1.0V, ISGDATA1[4:0]=00000B	37.5	50.0	50.0	
		5V	Ta=25°C, V_{ISINK1}=3.0V, ISGDATA1[4:0]=11111B	178	205	231	
		—	Ta=-40°C ~85°C, V_{ISINK1}=1.0V~4.5V, ISGDATA1[4:0]=11111B	168	205	242	
		—	Ta=-40°C ~85°C, V_{ISINK1}=0.7V~1.0V, ISGDATA1[4:0]=11111B	154	205	205	



V_{SINK0} VS. I_{SINK0}

I_{SINK0} Characteristic Curve



V_{SINK1} VS. I_{SINK1}

I_{SINK1} Characteristic Curve

16-bit Vioce D/A Converter Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DAC}	Additional Current for D/A Converter Enable With Buffer	3V	—	—	2.5	4.0	mA
		5V	—	—	3.5	5.5	
$I_{STB(DAC)}$	Standby Current	5V	DACEN=0	—	—	1	μA
THD+N	Total Harmonic Distortion+Noise(Note)	3V	10kΩ load	—	-55	—	dB
V_{OUT}	Output Voltage Range	5V	No load	0.01	—	0.99	V_{DD}
t_{DACS}	D/A Converter Circuit Turn on Stable Time	5V	—	—	—	0.2	ms

Note: Sin wave input @ 1kHz, -6dBFS.

Power Line Transceiver Electrical Characteristics

Comparator Characteristics

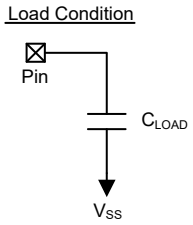
Ta=25°C

All the parameters are measured under condition of comparator input voltage=(V_{DD}-1)/2 and remain constant.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Comparator Operating Voltage	—	—	2.2	—	5.5	V
I_{CMP}	Additional Current for Comparator Enable	3V	No load, PLTCmIS[1:0]=00B (m=0, 1)	—	1	5	μA
		5V		—	1	5	
		3V	No load, PLTCmIS[1:0]=01B (m=0, 1)	—	14	30	
		5V		—	14	30	
		3V	No load PLTCmIS[1:0]=10B (m=0, 1)	—	36	65	
		5V		—	36	65	
		3V	No load, PLTCmIS[1:0]=11B (m=0, 1)	—	58	110	
		5V		—	58	110	
V_{OS}	Comparator Input Offset Voltage	3V	Without calibration (PLTCmOF[4:0]=10000B, PLTCmIS[1:0]=00B, m=0, 1)	-10	—	+10	mV
		5V		-10	—	+10	
		3V	With calibration	-4	—	+4	
		5V		-4	—	+4	
V_{CM}	Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}-1	V
t_{RP}	Response Time	3V	With 10mV overdrive(Note), No debounce, PLTCmIS[1:0]=00B (m=0, 1)	—	—	40	μs
		5V		—	—	40	
		3V	With 10mV overdrive(Note), No debounce, PLTCmIS[1:0]=01B (m=0, 1)	—	—	4	
		5V		—	—	4	
		3V	With 10mV overdrive(Note), No debounce, PLTCmIS[1:0]=10B (m=0, 1)	—	—	2	
		5V		—	—	2	
		3V	With 10mV overdrive(Note), No debounce, PLTCmIS[1:0]=11B (m=0, 1)	—	—	1.5	
		5V		—	—	1.5	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{HYS}	Hysteresis	3V	PLTCmHYS[1:0]=00B, PLTCmIS[1:0]=00B (m=0, 1)	0	0	5	mV
		5V		0	0	5	
		3V	PLTCmHYS[1:0]=01B, PLTCmIS[1:0]=01B (m=0, 1)	20	40	65	
		5V		20	40	65	
		3V	PLTCmHYS[1:0]=10B, PLTCmIS[1:0]=10B (m=0, 1)	50	100	150	
		5V		50	100	150	
		3V	PLTCmHYS[1:0]=11B, PLTCmIS[1:0]=11B (m=0, 1)	80	160	240	
		5V		80	160	240	

Note: Load Condition: C_{LOAD}=50pF



Operational Amplifier Characteristics

V_{DD}=5V, Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
I_{OPA}	Operating Current	—	PLTABW=0, no load	—	80	128	μA
			PLTABW=1, no load	—	200	320	
V_{OS}	Input Offset Voltage	—	Without calibration (PLTAOF[5:0]=100000B)	-15	—	15	mV
			With calibration	-2	—	2	
I_{OS}	Input Offset Current	—	V_{IN}=1/2 V_{CM}	—	1	10	nA
V_{CM}	Common Mode Voltage Range	—	PLTABW=0 or 1	V_{SS}	—	V_{DD} -1.4	V
PSRR	Power Supply Rejection Ratio	—	PLTABW=0 or 1	50	70	—	dB
CMRR	Common Mode Rejection Ratio	—	PLTABW=0 or 1	50	80	—	dB
A_{OL}	Open Loop Gain	—	PLTABW=0 or 1	60	80	—	dB
SR	Slew Rate	—	R_{LOAD}=1MΩ, C_{LOAD}=60pF, PLTABW=0	180	500	—	V/ms
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, PLTABW=1	600	1800	—	
GBW	Gain Bandwidth	—	R_{LOAD}=1MΩ, C_{LOAD}=60pF, PLTABW=0	400	600	—	kHz
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, PLTABW=1	1300	2000	—	
V_{OR}	Maximum Output Voltage Range	—	PLTABW=0 or 1, R_{LOAD}=5kΩ to V_{DD}/2	V_{SS} +210	—	V_{DD} -230	mV
I_{SC}	Output Short Circuit Current	—	R_{LOAD}=5.1Ω, PLTABW=0 or 1	±8.5	±20	—	mA

Note: These parameters are characterized but not tested.

V_{DD}=2.2V~5.5V, Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OPA}	Operating Current	—	PLTABW=0, no load	—	80	128	μA
			PLTABW=1, no load	—	200	320	
V_{OS}	Input Offset Voltage	—	Without calibration (PLTAOF [5:0]=100000B)	-15	—	15	mV
			With calibration	-2	—	2	
I_{OS}	Input Offset Current	—	V_{IN}=1/2 V_{CM}	—	1	10	nA
V_{CM}	Common Mode Voltage Range	—	PLTABW=0 or 1	V_{SS}	—	V_{DD} -1.4	V
PSRR	Power Supply Rejection Ratio	—	PLTABW=0 or 1	50	70	—	dB
CMRR	Common Mode Rejection Ratio	—	PLTABW=0 or 1	50	80	—	dB
A_{OL}	Open Loop Gain	—	PLTABW=0 or 1	60	80	—	dB
SR	Slew Rate	—	R_{LOAD}=1MΩ, C_{LOAD}=60pF, PLTABW=0	180	500	—	V/ms
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, PLTABW=1	600	1800	—	
GBW	Gain Bandwidth	—	R_{LOAD}=1MΩ, C_{LOAD}=60pF, PLTABW=0	250	600	—	kHz
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, PLTABW=1	800	2000	—	
V_{OR}	Maximum Output Voltage Range	—	PLTABW=0 or 1, R_{LOAD}=5kΩ to V_{DD}/2	V_{SS} +210	—	V_{DD} -230	mV
I_{SC}	Output Short Circuit Current	—	R_{LOAD}=5.1Ω, PLTABW=0 or 1	±2	±20	—	mA

Note: These parameters are characterized but not tested.

D/A Converter Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
V_{DACO}	Output Voltage Range	—	—	V_{SS}+0.1	—	V_{REF}-0.1	V
V_{REF}	Reference Voltage (DAC0 & DAC1)	—	PLTDA0REFS=1 or PLTDA1REFS=1	V_{SS}+0.4	—	V_{DD}-1.0	V
	Reference Voltage (DAC2)	—	—	2	—	V_{DD}	V
I_{DAC}	Additional Current for D/A Converter Enable (DAC0 & DAC1)	3V	PLTDACC[4:3]=00, PLTDACC[1:0]=01 or 10	—	3	6	μA
			PLTDACC[4:3]=00, PLTDACC[1:0]=11	—	6	12	
			V_{PLTREF}=1.6V, PLTDACC[4:3]=10, PLTDACC[1:0]=10	—	5.6	11.2	
			V_{PLTREF}=1.6V, PLTDACC[4:3]=11, PLTDACC[1:0]=11	—	7.2	14.4	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DAC}	Additional Current for D/A Converter Enable (DAC0 & DAC1)	5V	PLTDACC[4:3]=00, PLTDACC[1:0]=01 or 10	—	5	10	μA
			PLTDACC[4:3]=00, PLTDACC[1:0]=11	—	10	20	
			V_{PLTREF}=1.6V, PLTDACC[4:3]=10, PLTDACC[1:0]=10	—	6.6	13.2	
			V_{PLTREF}=1.6V, PLTDACC[4:3]=11, PLTDACC[1:0]=11	—	8.2	16.4	
	Additional Current for D/A Converter Enable (DAC2)	3V	—	—	—	360	μA
		5V	—	—	—	600	
t_{ST}	Settling Time	3V	C_{LOAD}=50pF	—	—	5	μs
		5V		—	—	5	
DNL	Differential Non-linearity	3V	V_{REF}=V_{DD}	-1	—	+1	LSB
		5V		-1	—	+1	
INL	Integral Non-linearity	3V	V_{REF}=V_{DD}	-1.5	—	+1.5	LSB
		5V		-1.5	—	+1.5	
Ro	R2R Output Resistor (DAC0 & DAC1)	3V	—	—	1000	—	kΩ
		5V	—	—	1000	—	
	R2R Output Resistor (DAC2)	3V	—	—	10	—	kΩ
		5V	—	—	10	—	
OSRR	Offset Error	3V	—	—	—	6	mV
		5V	—	—	—	10	
GERR	Gain Error	3V	—	—	—	12	mV
		5V	—	—	—	20	

Smoke Detector AFE Electrical Characteristics

V_{DD}=5V, Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OPA}	Operating Current	—	SDAmBW[1:0]=00B, no load	—	3.0	5.0	μA
			SDAmBW[1:0]=01B, no load	—	10	16	
			SDAmBW[1:0]=10B, no load	—	80	128	
			SDAmBW[1:0]=11B, no load	—	200	320	
V_{OS}	Input Offset Voltage	—	Without calibration (SDAmOF[5:0]=100000B)	-15	—	15	mV
			With calibration	-2	—	2	
V_{CM}	Common Mode Voltage Range	—	SDAmBW[1:0]=00, 01, 10, 11	V_{SS}	—	V_{DD} -1.4	V
PSRR	Power Supply Rejection Ratio	—	SDAmBW[1:0]=00, 01, 10, 11	50	70	—	dB
CMRR	Common Mode Rejection Ratio	—	SDAmBW[1:0]=00, 01, 10, 11	50	80	—	dB
A_{OL}	Open Loop Gain	—	SDAmBW[1:0]=00, 01, 10, 11	60	80	—	dB

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
SR	Slew Rate	—	R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=00	0.5	1.5	—	V/ms
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=01	5	15	—	
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=10	180	500	—	
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=11	600	1800	—	
GBW	Gain Bandwidth	—	R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=00	2.5	5.0	—	kHz
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=01	20	40	—	
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=10	400	600	—	
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=11	1300	2000	—	
V_{OR}	Maximum Output Voltage Range	—	SDAmBW[1:0]=00, 01, R_{LOAD}=5kΩ to V_{DD}/2	V_{SS} +140	—	V_{DD} -160	mV
		—	SDAmBW[1:0]=10, 11, R_{LOAD}=5kΩ to V_{DD}/2	V_{SS} +120	—	V_{DD} -140	mV
I_{SC}	Output Short Circuit Current	—	R_{LOAD}=5.1Ω, SDAmBW[1:0]=00, 01	±6	±12	—	mA
		—	R_{LOAD}=5.1Ω, SDAmBW[1:0]=10, 11	±10	±20	—	mA

Note: These parameters are characterized but not tested.

V_{DD}=2.2V~5.5V, Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OPA}	Operating Current	—	SDAmBW[1:0]=00B, no load	—	2.5	4.0	μA
			SDAmBW[1:0]=01B, no load	—	10	16	
			SDAmBW[1:0]=10B, no load	—	80	128	
			SDAmBW[1:0]=11B, no load	—	200	320	
V_{OS}	Input Offset Voltage	3V	Without calibration (SDAmOF[5:0]=100000B)	-15	—	15	mV
		5V		-15	—	15	
		3V	With calibration	-2	—	2	
		5V		-2	—	2	
I_{OS}	Input Offset Current	—	Ta=25°C, V_{DD}=3.3V, V_{IN}=0.3V	—	50	100	pA
V_{CM}	Common Mode Voltage Range	—	SDAmBW[1:0]=00, 01, 10, 11	V_{SS}	—	V_{DD} -1.4	V
PSRR	Power Supply Rejection Ratio	—	SDAmBW[1:0]=00, 01, 10, 11	50	70	—	dB
CMRR	Common Mode Rejection Ratio	—	SDAmBW[1:0]=00, 01, 10, 11	50	80	—	dB
A_{OL}	Open Loop Gain	—	SDAmBW[1:0]=00, 01, 10, 11	60	80	—	dB
SR	Slew Rate	—	R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=00B	0.5	1.5	—	V/ms
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=01B	5	15	—	
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=10B	180	500	—	
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=11B	600	1800	—	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
GBW	Gain Bandwidth	—	R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=00B	2	5	—	kHz
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=01B	15	40	—	
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=10B	250	600	—	
			R_{LOAD}=1MΩ, C_{LOAD}=60pF, SDAmBW[1:0]=11B	800	2000	—	
V_{OR}	Maximum Output Voltage Range	—	SDAmBW[1:0]=00, 01, R_{LOAD}=5kΩ to V_{DD}/2	V_{SS} +140	—	V_{DD} -160	mV
		—	SDAmBW[1:0]=10, 11, R_{LOAD}=5kΩ to V_{DD}/2	V_{SS} +120	—	V_{DD} -140	mV
I_{SC}	Output Short Circuit Current	—	R_{LOAD}=5.1Ω, SDAmBW[1:0]=00, 01	±1.2	±12	—	mA
		—	R_{LOAD}=5.1Ω, SDAmBW[1:0]=10, 11	±2	±20	—	mA
e_n	Equivalent Input Voltage Noise	—	SDAmBW [1:0]=00B, f=1kHz	—	250	—	nV/√Hz
		—	SDAmBW [1:0]=01B, f=1kHz f=10kHz	—	140 90	—	
		—	SDAmBW [1:0]=10B, f=1kHz f=10kHz f=100kHz	—	80 40 30	—	
		—	SDAmBW [1:0]=11B, f=1kHz f=10kHz f=100kHz f=1000kHz	—	80 40 20 20	—	

Note: These parameters are characterized but not tested.

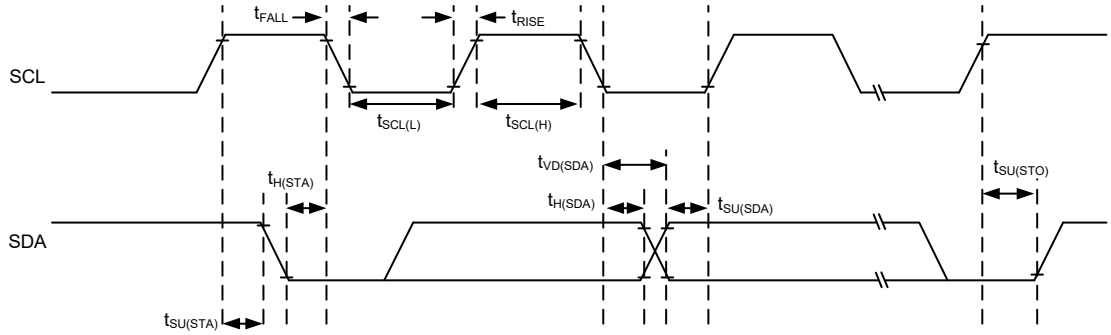
I²C Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{I2C}	I²C Standard Mode (100kHz) f_{SYS} Frequency (Note)	—	No clock debounce	2	—	—	MHz
		—	2 system clock debounce	4	—	—	MHz
		—	4 system clock debounce	4	—	—	MHz
	I²C Fast Mode (400kHz) f_{SYS} Frequency (Note)	—	No clock debounce	4	—	—	MHz
		—	2 system clock debounce	8	—	—	MHz
		—	4 system clock debounce	8	—	—	MHz
f_{SCL}	SCL Clock Frequency	3V/5V	Standard mode	—	—	100	kHz
			Fast mode	—	—	400	
$t_{SCL(H)}$	SCL Clock High Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
$t_{SCL(L)}$	SCL Clock Low Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
t_{FALL}	SCL and SDA Fall Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{RISE}	SCL and SDA Rise Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
$t_{SU(SDA)}$	SDA Data Setup Time	3V/5V	Standard mode	0.25	—	—	μs
			Fast mode	0.1	—	—	
$t_{H(SDA)}$	SDA Data Hold Time	3V/5V	—	0.1	—	—	μs
$t_{VD(SDA)}$	SDA Data Valid Time	3V/5V	—	—	—	0.6	μs
$t_{SU(STA)}$	Start Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	
$t_{H(STA)}$	Start Condition Hold Time	3V/5V	Standard mode	4.0	—	—	μs
			Fast mode	0.6	—	—	
$t_{SU(STO)}$	Stop Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	

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



I²C Timing Diagram

Buzzer Driver Electrical Characteristics

BDV_{DD}=3V, C1=47μF, L1=10μH, C2=2.2μF and Ta=25 °C, unless otherwise specified

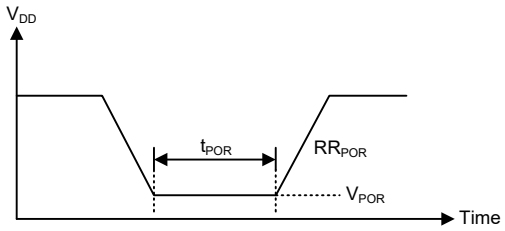
Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
Power Supply						
BDV_{DD}	Supply voltage	—	2.2	—	5.5	V
I_Q	Quiescent Current	BDV_{DD}=3V, No load, BWDT ON, other circuits off	—	—	2	μA
I_{IN}	Operating Current	BDV_{DD}=3V, No load, Boost on, Regulator on, 2-pin buzzer mode, f_{ENCLK}=4kHz	—	2	4	mA
I_{IL}	Input Leakage Current	MODE=BDV_{DD} or BDV_{SS}	—	—	±0.1	μA
		ENB=BDV_{DD}	—	—	-0.1	μA
		WDI, ENCLK=BDV_{SS}	—	—	0.1	μA
V_{IH}	High-Level Input Voltage	WDI, ENB, MODE, ENCLK	0.8BDV_{DD}	—	BDV_{DD}	V
V_{IL}	Low-Level Input Voltage	WDI, ENB, MODE, ENCLK	0	—	0.2BDV_{DD}	V
R_{PH}	Pull-High Resistor	BDV_{DD}=3V, ENB	—	200	—	kΩ
R_{PL}	Pull-Low Resistor	BDV_{DD}=3V, ENCLK, WDI	—	1	—	MΩ
BWDT						
t_{WDI}	WDI Period	BDV_{DD}=2.2V~5.5V, Ta=-40~85°C	10.3	11.5	14.3	s
t_{WDTP}	BWDT Reset Pulse Width	—	50	—	—	ns

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
Boost Converter						
V_{OUT}	Output Voltage	—	8.1	9.0	9.9	V
I_{OUT}	Output Current	—	50	—	—	mA
I_{OCP}	Over Current Protection	—	—	0.95	—	A
f_{SW}	Switching Frequency	—	0.8	1	1.2	MHz
Regulator						
V_{V5}	Regulator Output Voltage	—	4.9	5.2	5.5	V
I_{V5}	Output Current	—	5	—	—	mA
Buzzer						
I_{IL}	Leakage Current	Buzzer off, FB=BDV_{SS}	—	—	1	μA
		Buzzer off, FB=V_{OUT}	—	-2.25	—	
V_{IH}	High-Level Input Voltage	FB	0.7V_{OUT}	—	V_{OUT}	—
V_{IL}	Low-Level Input Voltage	FB	BDV_{SS}	—	0.3V_{OUT}	—
I_{OH}	VB, VS Source Current	V_{OUT}=9V, V_{OH}=0.9V_{OUT}	-70	-90	—	mA
I_{OL}	VB, VS Sink Current	V_{OUT}=9V, V_{OH}=0.1V_{OUT}	70	90	—	mA
t_{PLH}	VB, VS Output Delay Time	C_{LOAD}=10nF, 2-pin buzzer mode	—	1	1.5	μs
t_{PHL}	VB, VS Output Delay Time	C_{LOAD}=10nF, 2-pin buzzer mode	—	1	1.5	μs

Power-on Reset Characteristics

Ta=-40°C~85°C

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



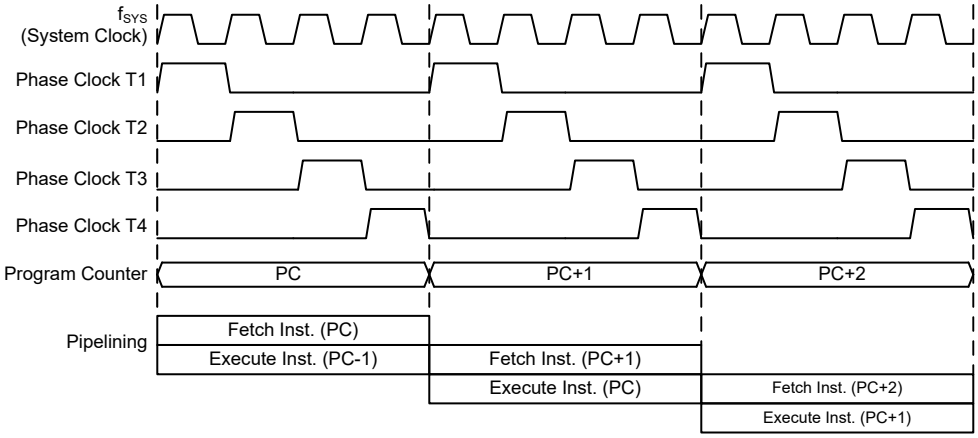
System Architecture

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

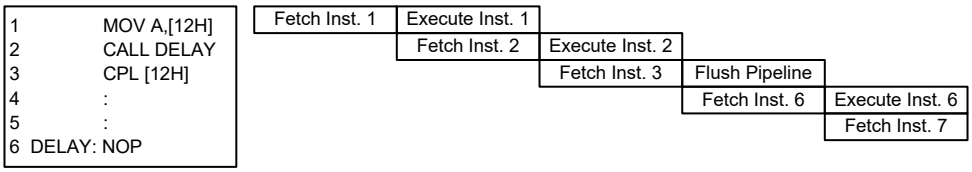
Clocking and Pipelining

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

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



System Clocking and Pipelining



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demands a jump to a non-consecutive Program Memory address. For the device with a program memory capacity in excess of 8K words, the program memory high byte address must be setup by selecting a certain program memory bank which is implemented using the program memory bank pointer bit, PBP0. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

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

Program Counter

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

Program Counter Read Registers

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

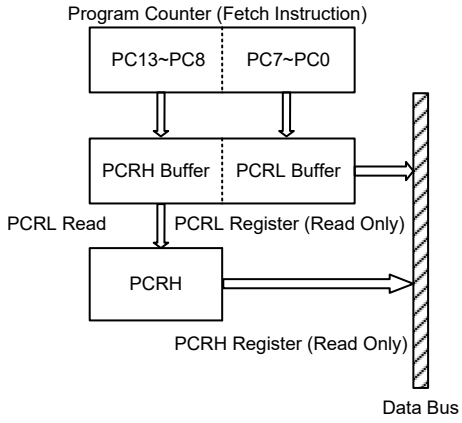
The following example shows how to read the current program execution address. When the current program execution address is 123H, the steps to execute the instructions are as follows:

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

MOV A, PCRH → the ACC value is 01H.

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

LMOV A, PCRH → the ACC value is 01H.



• **PCRL Register**

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

Bit 7~0 **D7~D0**: Program Counter Read Low byte register bit 7 ~ bit 0

• **PCRH Register**

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

Bit 7~6 Unimplemented, read as "0"

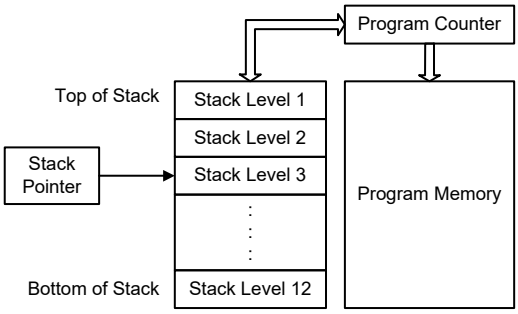
Bit 5~0 **D13~D8**: Program Counter Read High byte register bit 5 ~ bit 0

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 12 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, STKPTR[3:0]. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



• **STKPTR Register**

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

Bit 7 **OSF**: Stack overflow flag
0: No stack overflow occurred
1: Stack overflow occurred
When the stack is full and a CALL instruction is executed or when the stack is empty and a RET instruction is executed, the OSF bit will be set high. The OSF bit is cleared only by software and cannot be reset automatically by hardware.

Bit 6~4 Unimplemented, read as "0"

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

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

(1) When the CALL subroutine instruction is executed 13 times continuously and the RET instruction is not executed during the period, the corresponding changes of the STKPTR[3:0] and OSF bits are as follows:

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

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

(3) When the stack is empty, the RET instruction is executed 12 times continuously, the corresponding changes of the STKPTR[3:0] and OSF bits are as follows:

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

Arithmetic and Logic Unit – ALU

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

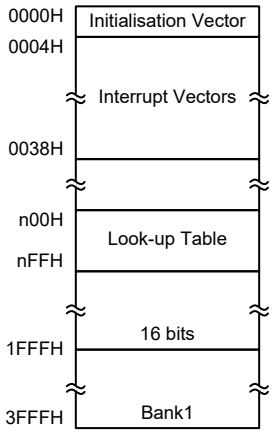
- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
  LAND, LOR, LXOR, LANDM, LORM, LXORM, LCPL, LCPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,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

Flash Program Memory

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

Structure

The Program Memory has a capacity of 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.



Program Memory Structure

Special Vectors

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

Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the corresponding table read instruction such as "TABRD [m]" 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.

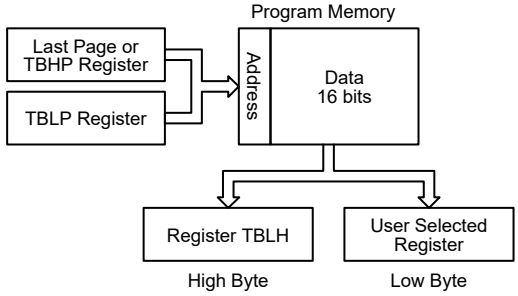


Table Program Example

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 is located in ROM Bank 1 and refers to the start address of the last page within the 16K words Program Memory. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "3F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address specified by TBLP and TBHP if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

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

Table Read Program Example

```
rombank 1 code1
ds .section 'data'
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,3Fh         ; initialise high table pointer
mov tbhp,a        ; it is not necessary to set tbhp if executing tabrdl or ltabrdl
:
:
tabrd tempreg1    ; transfers value in table referenced by table pointer data at
                  ; program memory address "3F06H" 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 "3F05H" transferred to tempreg2
                  ; and TBLH
                  ; in this example the data "1AH" is transferred to tempreg1 and
                  ; data "0FH" to register tempreg2
                  ; the value "00H" will be transferred to the high byte register
                  ; TBLH
:
:
code1 .section 'code'
org 1F00h         ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

In Circuit Programming – ICP

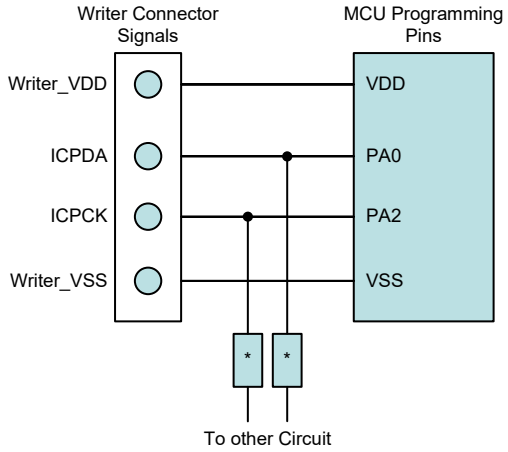
The provision of Flash Type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, a means of programming the microcontroller in-circuit has provided using a 4-pin interface. This provides manufacturers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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VDD	VDD	Power Supply
VSS	VSS	Ground

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

During the programming process, the user must take care of the ICPDA and ICPCK pins for data and clock programming purposes to ensure that no other outputs are connected to these two pins.



Note: * may be resistor or capacitor. The resistance of * must be greater than 1kΩ or the capacitance of * must be less than 1nF.

On Chip Debug Support – OCDS

An EV chip exists for the purposes of device emulation. This EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

In Application Programming – IAP

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

Flash Memory Read/Write Size

The Flash memory Erase and Write operations are carried out in a page format while the Read operation is carried out in a word format. The page size and write buffer size are both assigned with a capacity of 64 words. Note that the Erase operation should be executed before the Write operation is executed.

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

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

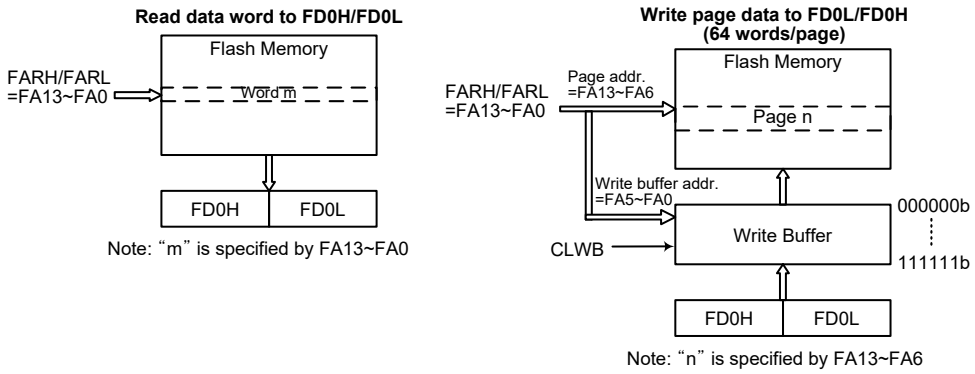
Operations	Format
Erase	64 words/page
Write	64 words/time
Read	1 word/time
Note: Page size=Write buffer size=64 words.	

IAP Operation Format

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 0000	00	xx xxxx
: :	: :	: :	: :
254	0011 1111	10	xx xxxx
255	0011 1111	11	xx xxxx

"x": don't care

Erase Page Number and Selection



Flash Memory IAP Read/Write Structure

Write Buffer

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

The write buffer size is 64 words corresponding to a page. The write buffer address is mapped to a specific Flash memory page specified by the memory address bits, FA13~FA6. The data written into the FD0L and FD0H registers will be loaded into the write buffer. When data is written into the high byte data register, FD0H, it will result in the data stored in the high and low byte data registers both being written into the write buffer. It will also cause the Flash memory address to be incremented by one, after which the new address will be loaded into the FARH and FARL address registers. When the Flash memory address reaches the page boundary, 111111b of a page with 64 words, the address will now not be incremented but stop at the last address of the page. At this point a new page address should be specified for any other erase/write operations.

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

IAP Flash Program Memory Registers

There are two address registers, four pairs of 16-bit data registers and three control registers, which are all located in Sector 1. Read and Write operations to the Flash memory are carried out using 16-bit data operations using the address and data registers and the control register. Several registers control the overall operation of the internal Flash Program Memory. The address registers are named FARL and FARH, the data registers are named FDnL and FDnH and the control registers are named FC0, FC1 and FC2. As all the IAP related registers are located in Sector 1, they can be addressed directly only using the corresponding extended instructions or can be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pairs and Indirect Addressing Register, IAR1 or IAR2.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	—	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	—	FA13	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8

Register Name	Bit							
	7	6	5	4	3	2	1	0
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Register List

- **FC0 Register**

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

Bit 7 **CFWEN**: Flash Memory Erase/Write function enable control
0: Flash memory erase/write function is disabled
1: Flash memory erase/write function has been successfully enabled

When this bit is cleared to 0 by application program, the Flash memory erase/write function is disabled. Note that this bit cannot be set high by application programs. Writing a "1" into this bit results in no action. This bit is used to indicate the Flash memory erase/write function status. When this bit is set to 1 by hardware, it means that the Flash memory erase/write function is enabled successfully. Otherwise, the Flash memory erase/write function is disabled if the bit is zero.

Bit 6~4 **FMOD2~FMOD0**: Flash memory Mode selection
000: Write Mode
001: Page Erase Mode
010: Reserved
011: Read Mode
100: Reserved
101: Reserved
110: Flash memory Erase/Write function Enable Mode
111: Reserved

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

Bit 3 **FWPEN**: Flash memory Erase/Write function enable procedure trigger
0: Erase/Write function enable procedure is not triggered or procedure timer times out
1: Erase/Write function enable procedure is triggered and procedure timer starts to count

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

Bit 2 **FWT**: Flash memory write initiate control
0: Do not initiate Flash memory write or indicating that a Flash memory write process has completed
1: Initiate Flash memory write process

This bit is set by software and cleared by hardware when the Flash memory write process has completed.

Bit 1 **FRDEN**: Flash memory read enable control
0: Flash memory read disable
1: Flash memory read enable

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

Bit 0 **FRD**: Flash memory read initiate control
0: Do not initiate Flash memory read or indicating that a Flash memory read process has completed
1: Initiate Flash memory read process
This bit is set by software and cleared by hardware when the Flash memory read process has completed.

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

• **FC1 Register**

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

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

• **FC2 Register**

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

Bit 7~1 Unimplemented, read as "0"
Bit 0 **CLWB**: Flash memory Write Buffer Clear control
0: Do not initiate a Write Buffer Clear process or indicating that a Write Buffer Clear process has completed
1: Initiate Write Buffer Clear process
This bit is set by software and cleared by hardware when the Write Buffer Clear process has completed.

• **FARL Register**

Bit	7	6	5	4	3	2	1	0
Name	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **FA7~FA0**: Flash Memory Address bit 7 ~ bit 0

• **FARH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	FA13	FA12	FA11	FA10	FA9	FA8
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 **FA13~FA8**: 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 **D7~D0**: The first Flash Memory data word bit 7 ~ bit 0

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

• **FD0H Register**

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

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

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

• **FD1L Register**

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

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

• **FD1H Register**

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

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

• **FD2L Register**

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

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

• **FD2H Register**

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

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

• **FD3L Register**

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

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

• **FD3H Register**

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

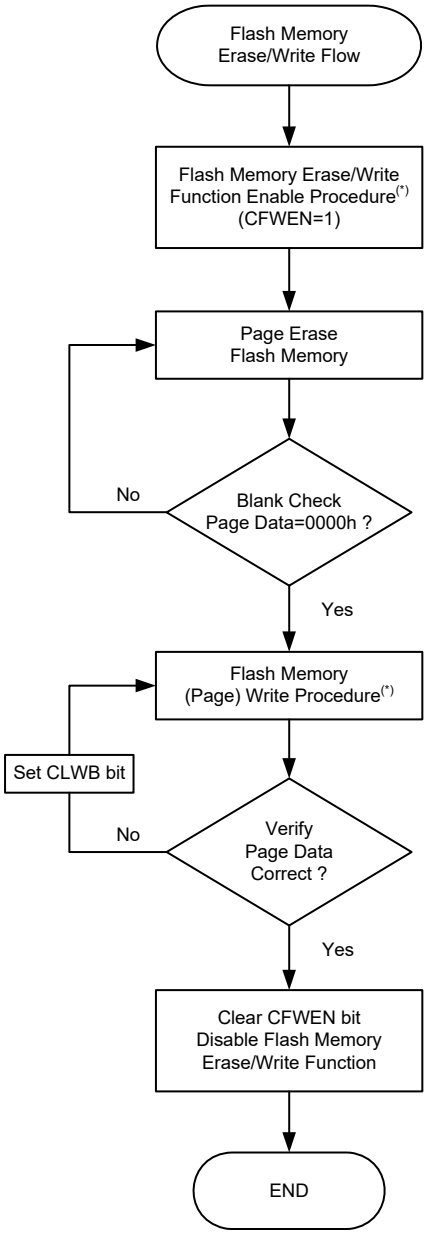
Bit 7~0 **D15~D8**: The fourth Flash Memory data word bit 15 ~ bit 8

Flash Memory Erase/Write Flow

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

Flash Memory Erase/Write Flow Descriptions

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



Flash Memory Erase/Write Flow

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

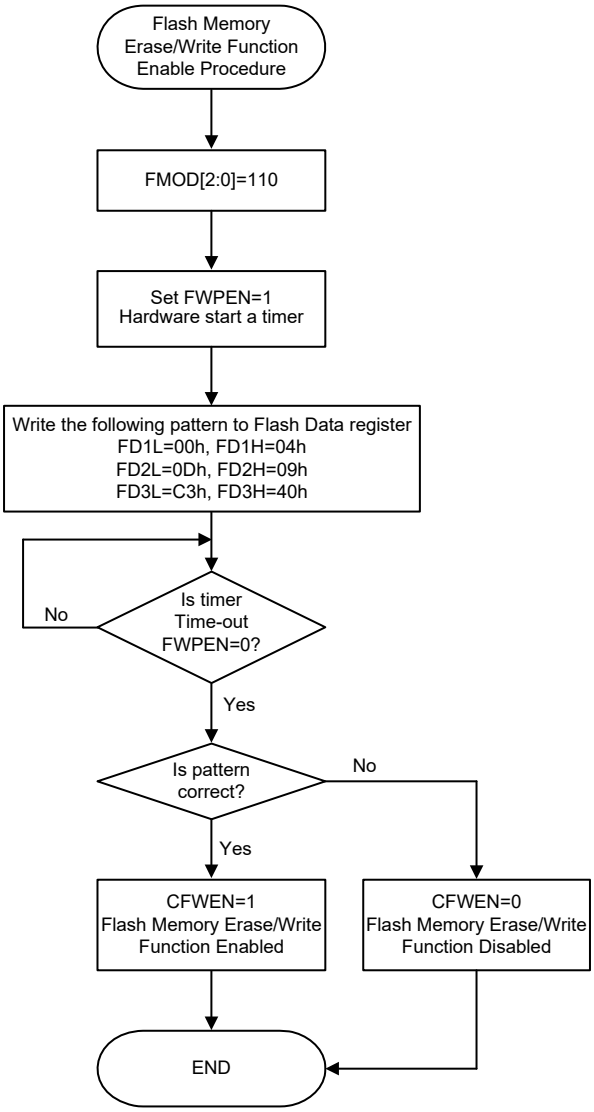
Flash Memory Erase/Write Function Enable Procedure

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

Flash Memory Erase/Write Function Enable Procedure Description

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

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



Flash Memory Erase/Write Function Enable Procedure

Flash Memory Write Procedure

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

The write buffer size is 64 words, known as a page, whose address is mapped to a specific Flash memory page specified by the memory address bits, FA13~FA6. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits specify.

Flash Memory Consecutive Write Description

The maximum amount of write data is 64 words for each write operation. The write buffer address will be automatically incremented by one when consecutive write operations are executed. The start address of a specific page should first be written into the FARL and FARH registers. Then the data word should be written into the FD0L register and then the FD0H register. At the same time the write buffer address will be incremented by one and then the next data word can be written into the FD0L and FD0H registers for the next address without modifying the address register pair, FARH and FARL. When the write buffer address reaches the page boundary the address will not be further incremented but will stop at the last address of the page.

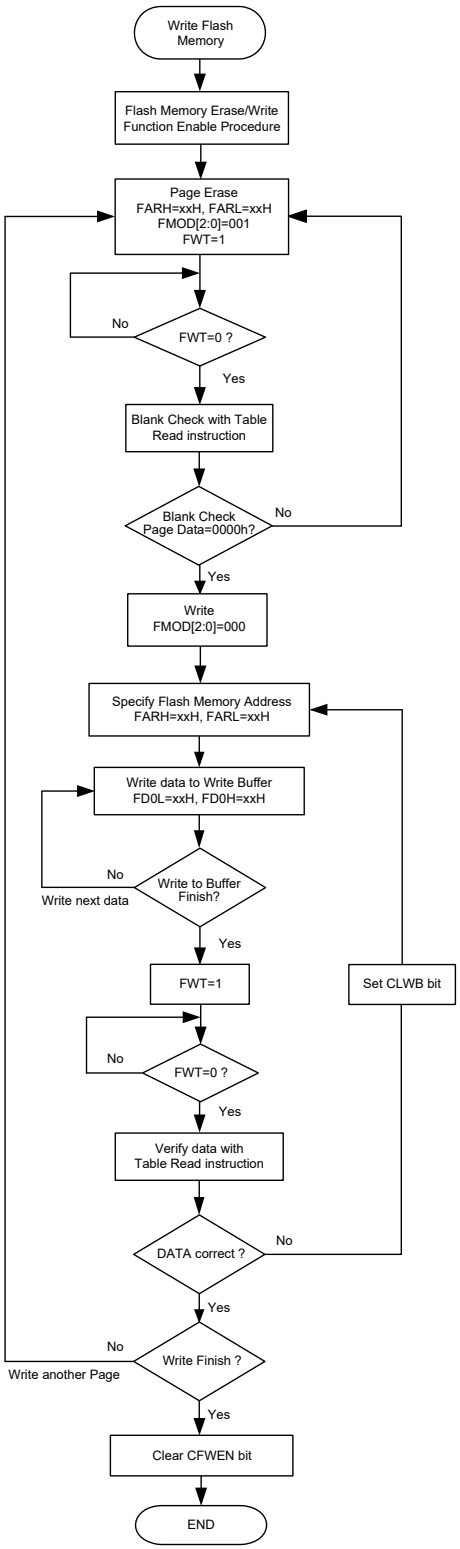
1. Activate the “Flash Memory Erase/Write function enable procedure”. Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the “Flash Memory Erase/Write function enable procedure” for more details.
2. Set the FMOD field to “001” to select the erase operation. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.

   Go to step 2 if the erase operation is not successful.

   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to “000” to select the write operation.
5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L and FD0H registers within a page as specified by their consecutive addresses. The maximum written data number is 64 words.
6. Set the FWT bit high to write the data words from the write buffer to the Flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

   If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 5.

   Go to step 8 if the write operation is successful.
8. Clear the CFWEN bit low to disable the Flash memory erase/write function.



Flash Memory Consecutive Write Procedure

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

Flash Memory Non-consecutive Write Description

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

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

1. Activate the "Flash Memory Erase/Write function enable procedure". Check the CFWEN bit value and then execute the erase/write operation if the CFWEN bit is set high. Refer to the "Flash Memory Erase/Write function enable procedure" for more details.
2. Set the FMOD field to "001" to select the erase operation. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.

   Go to step 2 if the erase operation is not successful.

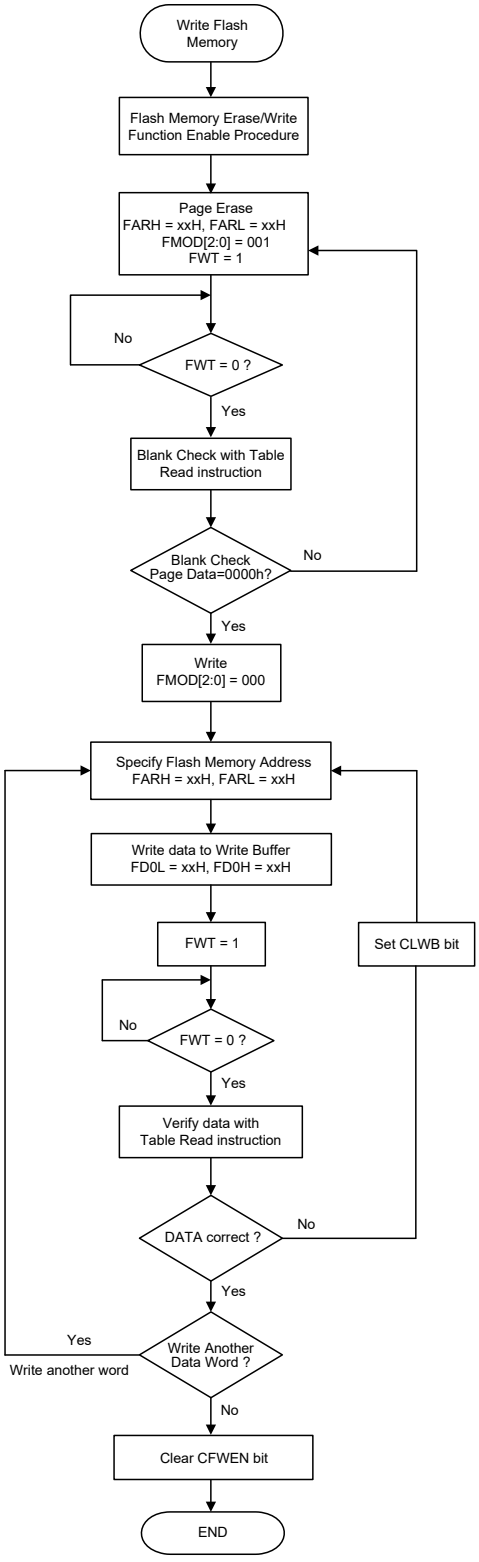
   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to "000" to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
6. Set the FWT bit high to transfer the data word from the write buffer to the Flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

   If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 5.

   Go to step 8 if the write operation is successful.
8. Setup the desired address ADDR2 in the FARH and FARL registers. Write the desired data word DATA2 first into the FD0L register and then into the FD0H register.
9. Set the FWT bit high to transfer the data word from the write buffer to the Flash memory. Wait until the FWT bit goes low.
10. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

    If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 8.

    Go to step 11 if the write operation is successful.
11. Clear the CFWEN bit low to disable the Flash memory erase/write function.



Flash Memory Non-consecutive Write Procedure

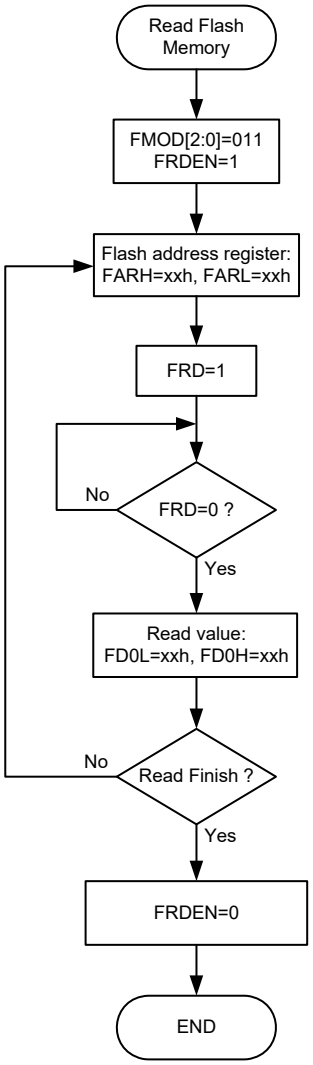
Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.
2. It will take a typical time of 2.2ms for the FWT bit state changing from high to low.

Important Points to Note for Flash Memory Write Operations

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

Flash Memory Read Procedure

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



Flash Memory Read Procedure

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

Data Memory

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

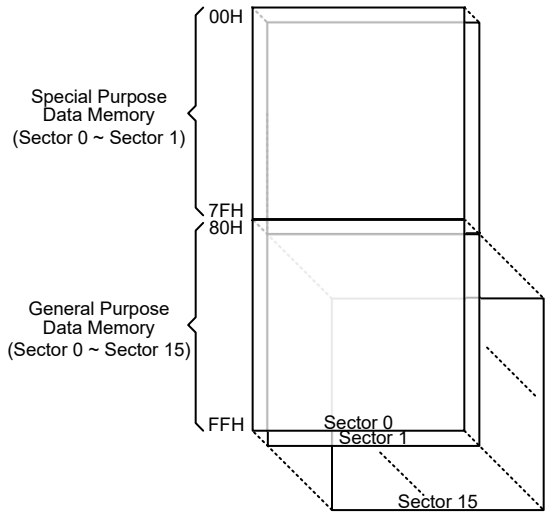
Categorized into two types, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some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

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

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 two types, the special Purpose Data Memory and the General Purpose Data Memory. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH.

Special Purpose Data Memory	General Purpose Data Memory	
Located Sectors	Capacity	Sector: Address
0, 1	2048×8	0: 80H~FFH 1: 80H~FFH : 15: 80H~FFH



Data Memory Structure

Data Memory Addressing

For the device that supports the extended instructions, there is no Bank Pointer for Data Memory addressing. The Bank Pointer, PBP, is only available for Program Memory. The desired Data Memory 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 12 valid bits, the high byte indicates a sector and the low byte indicates a specific address within the sector.

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0	Sector 1
00H	IAR0	FC0
01H	MP0	FC1
02H	IAR1	FC2
03H	MP1L	FARL
04H	MP1H	FARH
05H	ACC	FD0L
06H	PCL	FD0H
07H	TBLP	FD1L
08H	TBLH	FD1H
09H	TBHP	FD2L
0AH	STATUS	FD2H
0BH	PBP	FD3L
0CH	IAR2	FD3H
0DH	MP2L	
0EH	MP2H	VBGRC
0FH	RSTFC	ORMC
10H	SCC	LVDC
11H	HIRCC	TLVRC
12H	HXTC	
13H	LXTC	CRCCR
14H	PA	CRCIN
15H	PAC	CRCDL
16H	PAPU	CRCDH
17H	PAWU	
18H	PB	IECC
19H	PBC	
1AH	PBPU	STM1C0
1BH	PC	STM1C1
1CH	PCC	STM1DL
1DH	PCPU	STM1DH
1EH	PD	STM1AL
1FH	PDC	STM1AH
20H	PDPU	
21H	WDTC	PTM1C0
22H	SDSW0	PTM1C1
23H	SDSW1	PTM1DL
24H	SDPGAC0	PTM1DH
25H	SDPGAC1	PTM1AL
26H	SDA0C	PTM1AH
27H	SDA0VOS	PTM1RPL
28H	SDA1C	PTM1RPH
29H	SDA1VOS	
2AH	SADC0	PTM2C0
2BH	SADC1	PTM2C1
2CH	SADC2	PTM2DL
2DH	SADOL	PTM2DH
2EH	SADOH	PTM2AL
2FH		PTM2AH
30H		PTM2RPL
31H		PTM2RPH
32H		
33H		SLEDC0
34H	PLTSW	SLEDC1
35H	PLTDACC	IOSMTC
36H	PLTDA0L	PMPS0
37H	PLTDA1L	PMPS1
38H	PLTDA2L	
39H	PLTC0C	PCRL
3AH	PLTC0VOS	PCRH
3BH	PLTC1C	
3CH	PLTC1VOS	STKPTR
3DH	PLTCHYC	
3EH	PLTAC	EEA
3FH	PLTAVOS	EED

	Sector 0	Sector 1
40H	PLTDICC1	EEC
41H	PLTDICC0	
42H	INTEG	PAS0
43H	INTC0	PAS1
44H	INTC1	PBS0
45H	INTC2	PBS1
46H	INTC3	PCS0
47H	MFI0	PCS1
48H	MFI1	PDS0
49H	MFI2	PDS1
4AH	MFI3	IFS0
4BH	MFI4	IFS1
4CH	MFI5	
4DH		ISGENC
4EH		ISGDATA0
4FH		ISGDATA1
50H	PSCR	
51H	TB0C	
52H	TB1C	
53H	SIMC0	
54H	SIMC1	
55H	SIMD	
56H	SIMA/SIMC2	
57H	SIMTOC	
58H		
59H	U0SR	
5AH	U0CR1	
5BH	U0CR2	
5CH	U0CR3	
5DH	BRDH0	
5EH	BRDL0	
5FH	UFCR0	
60H	TXR_RXR0	
61H	RxCNT0	
62H	U1SR	
63H	U1CR1	
64H	U1CR2	
65H	U1CR3	
66H	BRDH1	
67H	BRDL1	
68H	UFCR1	
69H	TXR_RXR1	
6AH	RxCNT1	
6BH		
6CH	STM0C0	
6DH	STM0C1	
6EH	STM0DL	
6FH	STM0DH	
70H	STM0AL	
71H	STM0AH	
72H	PTM0C0	
73H	PTM0C1	
74H	PTM0C2	
75H	PTM0DL	
76H	PTM0DH	
77H	PTM0AL	
78H	PTM0AH	
79H	PTM0BL	
7AH	PTM0BH	
7BH	PTM0RPL	
7CH	PTM0RPH	
7DH	DAH	
7EH	DAL	
7FH	DACC	

: Unused, read as 00H

: Reserved, cannot be changed

Special Purpose Data Memory Structure

Special Function Register Description

Most of the Special Function Register details will be described in the relevant functional 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 extended 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                 ; increase 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               ; increase 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 device the program memory is divided into two 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-consecutice program memory address which is located in a certain bank selected by the program memory bank pointer bits will occur.

• **PBP Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **PBP0**: Program memory bank pointer bit 0
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 Byte Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Option Memory Mapping Register – ORMC

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

To successfully enable the Option Memory Mapping function, the specific pattern of 55H and AAH must be written into the ORMC register in two consecutive instruction cycles. It is therefore recommended that the global interrupt bit EMI should first be cleared before writing the specific pattern, and then set high again at a proper time according to users' requirements after the pattern is successfully written. An internal timer will be activated when the pattern is successfully written. The

mapping operation will be automatically finished after a period of $4\times t_{LIRC}$. Therefore, users should read the data in time, otherwise the Option Memory Mapping function needs to be restarted. After the completion of each consecutive write operation to the ORMC register, the timer will recount.

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

• **ORMC Register**

Bit	7	6	5	4	3	2	1	0
Name	ORMC7	ORMC6	ORMC5	ORMC4	ORMC3	ORMC2	ORMC1	ORMC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **ORMC7~ORMC0**: Option Memory Mapping specific pattern
When a specific pattern of 55H and AAH is written into this register, the Option Memory Mapping function will be enabled. Note that the register content will be cleared after the MCU is woke up from the IDLE/SLEEP mode.

Status Register – STATUS

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

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

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

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

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

• **STATUS Register**

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

"x": Unknown

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

Bit 6 **CZ**: The operational result of different flags for different instructions

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

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

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

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

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

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

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

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

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

The "C" flag is also affected by a rotate through carry instruction.

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 256×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address register, a data register 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, which are all located in Sector 1. The EEA and EED registers can be addressed directly using the extended instructions or can be read from or written to indirectly using the MP1H/ MP1L or MP2H/MP2L Memory Pointer pairs and Indirect Addressing Register, IAR1 or IAR2. However, the EEC regiser can only be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pair and Indirect Addressing Register, IAR1 or IAR2. Because the EEC control register is located at address 40H in Sector 1, the 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	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**

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 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 bit 7 ~ bit 0

• **EEC Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3 **WREN**: Data EEPROM Write Enable
0: Disable
1: Enable
This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2 **WR**: EEPROM Write Control
0: Write cycle has finished
1: Activate a write cycle
This is the Data EEPROM Write Control Bit and when set high by the application program will activate a write cycle. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.

Bit 1 **RDEN**: Data EEPROM Read Enable
0: Disable
1: Enable
This is the Data EEPROM Read Enable Bit which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.

Bit 0 **RD**: EEPROM Read Control
0: Read cycle has finished
1: Activate a read cycle
This is the Data EEPROM Read Control Bit and when set high by the application program will activate a read cycle. This bit will be automatically reset to zero by the hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

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

Reading Data from the EEPROM

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

Writing Data to the EEPROM

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

Write Protection

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

EEPROM Interrupt

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

Programming Considerations

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

When writing data, the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write cycle is executed and then set high again after a valid write activation procedure has completed. 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

In the following examples, except for some branch type instructions and the EEC register which must be addressed indirectly, since the EEA and EED registers are located in Sector 1, the relevant programs are demonstrated by direct addressing using extended instructions.

Reading data from the EEPROM – polling method

```
LMOV A, EEPROM_ADRES     ; user defined address
LMOV 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 if no more read operations are required
CLR  MP1H
LMOV A, EED              ; move read data to register
LMOV READ_DATA, A
```

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

Writing Data to the EEPROM – polling method

```
LMOV A, EEPROM_ADRES     ; user defined address
LMOV EEA, A
LMOV A, EEPROM_DATA      ; user defined data
LMOV EED, 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
CLR  EMI
SET  IAR1.3              ; set WREN bit, enable write operations
SET  IAR1.2              ; start Write Cycle - set WR bit - executed immediately
                         ; after setting WREN bit
SET  EMI
BACK:
SZ   IAR1.2              ; check for write cycle end
JMP  BACK
CLR  MP1H
```

Oscillators

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

Oscillator Overview

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

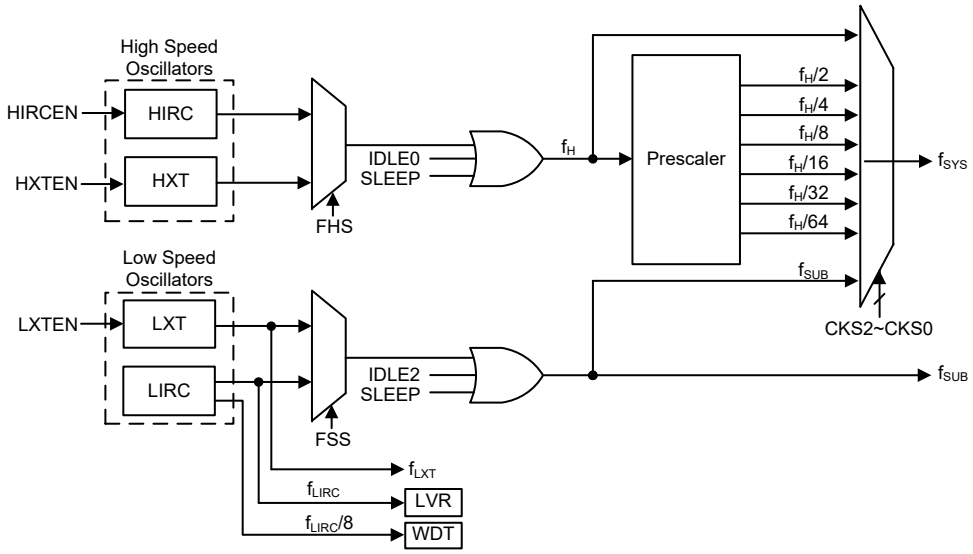
Type	Name	Frequency	Pins
Internal High Speed RC	HIRC	2/4/8MHz	—
External High Speed Crystal	HXT	400kHz~16MHz	OSC1/OSC2
Internal Low Speed RC	LIRC	32.768kHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2

Oscillator Types

System Clock Configurations

There are four oscillator sources, two high speed oscillators and two low speed oscillators. The high speed oscillators are the internal 2/4/8MHz RC oscillator, HIRC, and the external crystal/ceramic oscillator, HXT. The low speed oscillators are the internal 32.768kHz RC oscillator, LIRC, and the external 32.768kHz crystal oscillator, LXT. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

The actual source clock used for the low speed oscillator is chosen via the FSS bit in the SCC register while for the high speed oscillator the source clock is selected by the FHS bit in the SCC register. The frequency of the slow speed or high speed system clock is determined using the CKS2~CKS0 bits in the SCC register.

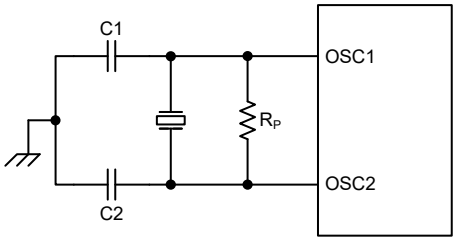


System Clock Configurations

External Crystal/Ceramic Oscillator – HXT

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

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



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

Crystal/Resonator Oscillator – HXT

HXT Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
16MHz	0 pF	0 pF
12MHz	0 pF	0 pF
8MHz	0 pF	0 pF
4MHz	0 pF	0 pF
1MHz	100 pF	100 pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

Internal High Speed RC Oscillator – HIRC

The internal high speed RC oscillator is one of the high frequency oscillator choices, which is selected by the FHS bit in the SCC register. It is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 2MHz, 4MHz and 8MHz, which are selected by HIRC1~HIRC0 bits in the HIRCC register. These bit must be setup to match the selected configuration option frequency to ensure that the HIRC frequency accuracy specified in the A.C. Characteristics is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

External 32.768kHz Crystal Oscillator – LXT

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

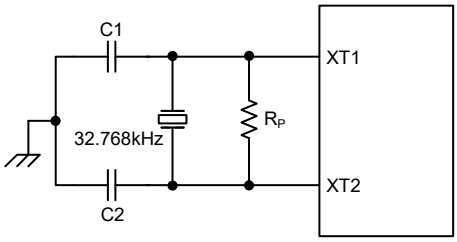
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE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

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

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

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

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



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

External LXT Oscillator

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

32.768kHz Crystal Recommended Capacitor Values

LXT Oscillator Speed Up Function

The LXT oscillator only works in the Speed Up Mode. The mode configuration is executed using the LXTSP bit in the register.

LXTSP	LXT Mode
0	Unused
1	Speed Up

When the LXTSP bit is set high, the LXT Speed Up Mode will be enabled. In the Speed Up Mode the LXT oscillator will power up and stabilise quickly. It is important to note that the LXT operating mode must be controlled before the LXT oscillator clock is selected as the system clock source. Once the LXT oscillator clock is selected as the system clock source using the CKS2~CKS0 bits and FSS bit in the SCC register, the LXTSP bit cannot be changed.

Internal 32.768kHz Oscillator – LIRC

The internal 32.768kHz 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 32.768kHz, requiring no external components for its implementation. In addition of supplying f_{LIRC} with the frequency of 32.768kHz, the clock source provides a divided version of $f_{LIRC}/8$ for the Watchdog Timer function. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

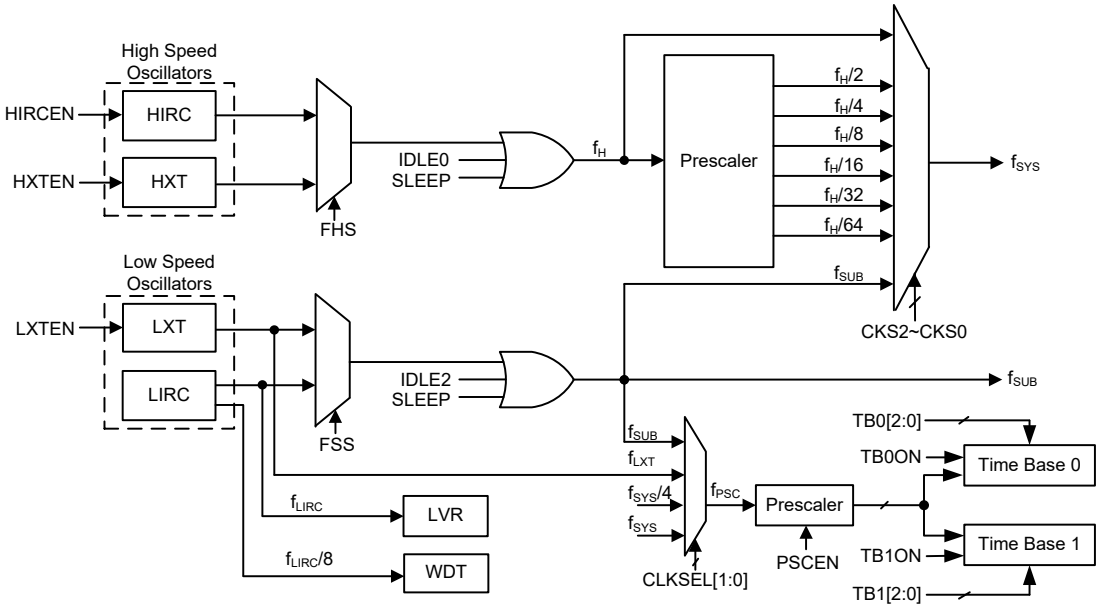
Operating Modes and System Clocks

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

System Clocks

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

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



Device Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillator will stop 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 Modes 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}	$f_{LIRC}/8$
		FHIDEN	FSIDEN	CKS2~CKS0					
FAST	On	x	x	000~110	f_H~$f_H/64$	On	On	On	On(2)
SLOW	On	x	x	111	f_{SUB}	On/Off(1)	On	On	On(2)
IDLE0	Off	0	1	000~110	Off	Off	On	On	On(2)
				111	On				
IDLE1	Off	1	1	xxx	On	On	On	On	On(2)
IDLE2	Off	1	0	000~110	On	On	Off	On	On(2)
				111	Off				
SLEEP	Off	0	0	xxx	Off	Off	Off	On(2)	On(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 $f_{LIRC}/8$ clock is always on as the WDT function is always enabled.
3. The f_{LXT} can be switched on or off by the LXTEN bit in the LXTC control register only. When the LXTEN bit is set to 1, the LXT will be turned on and provide the clock source for Time Base.

FAST Mode

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

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 determined by the FSS bit in the SCC register.

SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bit both are low. In the SLEEP mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral function will also be stopped. However, the $f_{LIRC}/8$ clock will continue to operate as the WDT function is always enabled.

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

The registers, SCC, HIRCC, HXTC and LXTC, are used to control the system clock and the corresponding oscillator configurations.

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

System Operating Mode Control Register List

- **SCC Register**

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

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

Bit 4 Unimplemented, read as "0"

Bit 3 **FHS**: High frequency clock selection
0: HIRC
1: HXT

Bit 2 **FSS**: Low frequency clock selection
0: LIRC
1: LXT

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

Bit 0 **FSIDEN**: Low Frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing a "HALT" instruction.

Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits, FHS bit or FSS bit. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.
Clock switching delay time=$4\times t_{SYS+}[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$, where $t_{Curr.}$ indicates the current clock period, $t_{Tar.}$ indicates the target clock period and t_{SYS} indicates the current system clock period.

• HIRCC Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection
00: 2MHz
01: 4MHz
10: 8MHz
11: 2MHz
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 high.
It is recommended that the HIRC frequency selected by these two bits should be same with the frequency determined by the configuration option to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator 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 (sink/source current is smaller)
1: HXT frequency > 10MHz (sink/source current is larger)
Note that this bit should be configured correctly according to the used HXT frequency. If HXTM=0 while the HXT frequency is larger than 10MHz, the oscillation performance at a low voltage condition may be not well. If HXTM=1 while the

HXT frequency is less than 10MHz, the oscillator frequency and the current may be obnormal.

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

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

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

Bit 0 **HXTEN**: HXT oscillator enable control
0: Disable
1: Enable

• LXTC Register

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

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LXTSP**: LXT speed up control
0: Disable – Unused
1: Enable – Speed up

This bit is used to control whether the LXT oscillator is operating in the speed up mode. It is important to note that this bit cannot be changed after the LXT oscillator is selected as the system clock source using the CKS2~CKS0 and FSS bits in the SCC register.

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

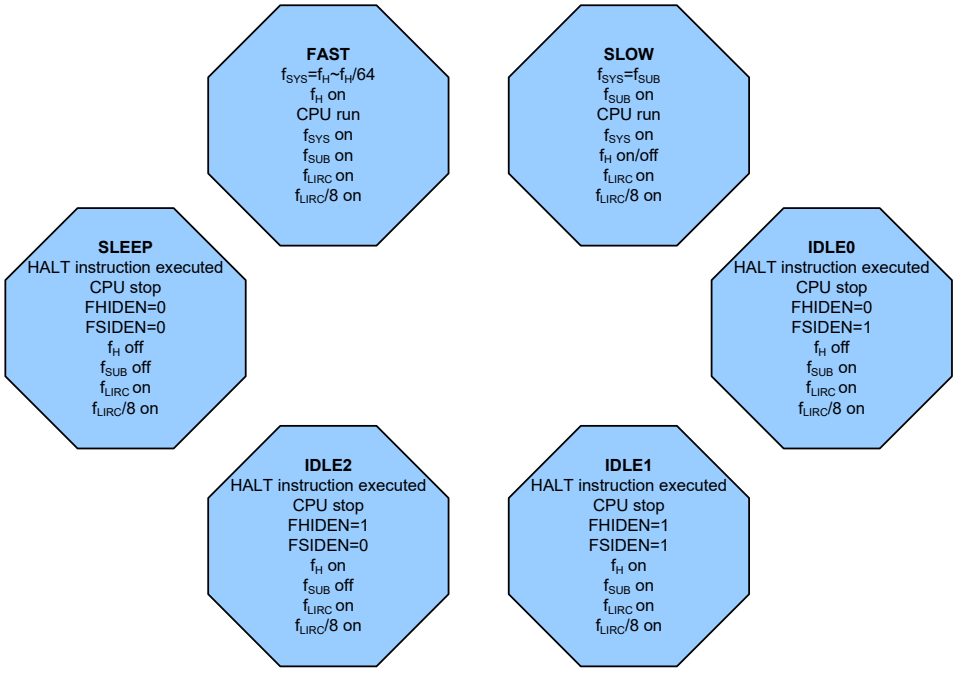
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set to 1 to enable the LXT oscillator, the LXTF bit will first be cleared to 0 and then set to 1 after the LXT oscillator is stable.

Bit 0 **LXTEN**: LXT oscillator enable control
0: Disable
1: Enable

Operating Mode Switching

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

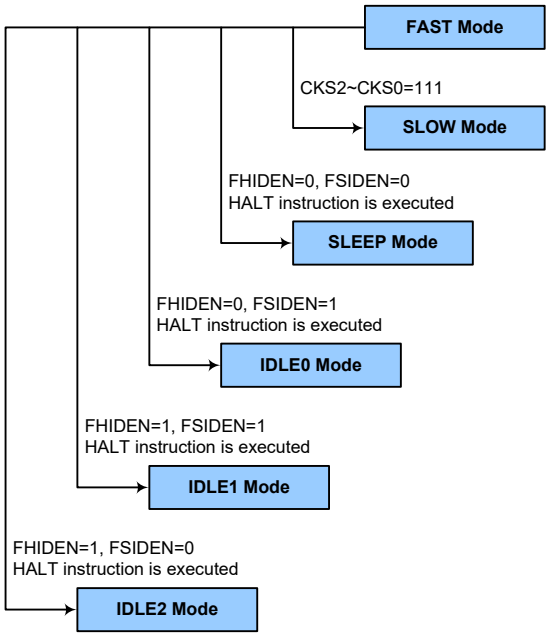
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



FAST Mode to SLOW Mode Switching

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

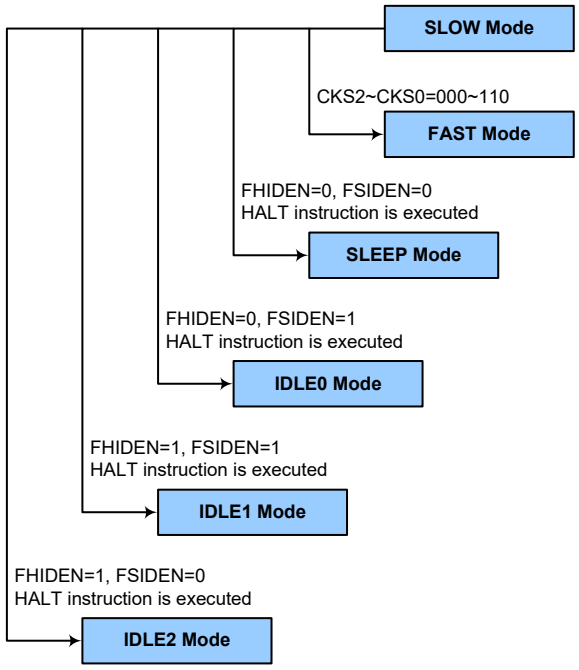
The SLOW Mode system clock is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires the selected oscillator to be stable before full mode switching occurs.



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~f_H/64.

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



Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

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

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

Entering the IDLE2 Mode

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

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

Standby Current Considerations

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

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

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

Wake-up

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

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

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

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

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a 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, $f_{LIRC}/8$, which is sourced from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32.768kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The Watchdog Timer source clock, $f_{LIRC}/8$, is then subdivided by a ratio of 2^5 to 2^{15} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period and WDT enable as well as 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 or 01010: Enable
Other values: Reset MCU
When these bits are changed by the environmental noise or software setting to reset the microcontroller, the reset operation will be activated after a delay time, t_{SRESET} and the WRF bit in the RSTFC register will be set high.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^5/(f_{LIRC}/8)=2^8/f_{LIRC}$
001: $2^7/(f_{LIRC}/8)=2^{10}/f_{LIRC}$
010: $2^9/(f_{LIRC}/8)=2^{12}/f_{LIRC}$
011: $2^{11}/(f_{LIRC}/8)=2^{14}/f_{LIRC}$
100: $2^{12}/(f_{LIRC}/8)=2^{15}/f_{LIRC}$
101: $2^{13}/(f_{LIRC}/8)=2^{16}/f_{LIRC}$
110: $2^{14}/(f_{LIRC}/8)=2^{17}/f_{LIRC}$
111: $2^{15}/(f_{LIRC}/8)=2^{18}/f_{LIRC}$
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	—	—	—	—	—	LVRF	—	WRF
R/W	—	—	—	—	—	R/W	—	R/W
POR	—	—	—	—	—	x	—	0

"x": unknown

Bit 7~3 Unimplemented, read as "0"

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

Bit 1 Unimplemented, read as "0"

Bit 0 **WRF**: WDTC register software reset flag
0: Not occurred
1: Occurred

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

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to offer the enable control of the Watchdog Timer. The WDT function will be enabled when the WE4~WE0 bits are equal to 10101B or 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.

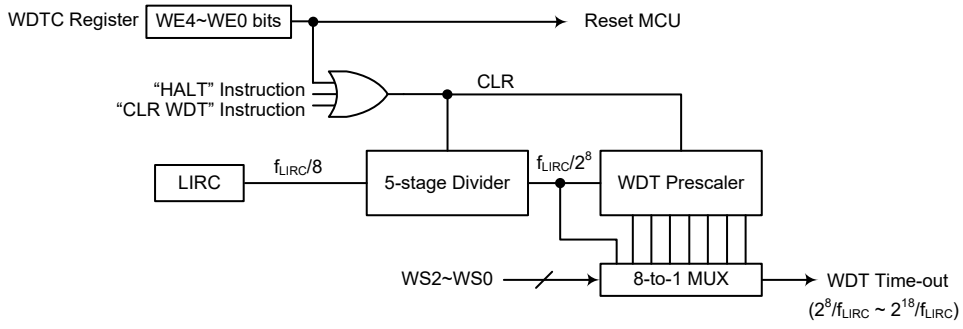
WE4~WE0 Bits	WDT Function
10101B or 01010B	Enable
Any other values	Reset MCU

Watchdog Timer Enable Control

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

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

The maximum time out period is when the 2^{15} division ratio is selected. As an example, with a $f_{LIRC}/8$ as its source clock, this will give a maximum watchdog period of around 8 seconds for the 2^{15} division ratio, and a minimum timeout of 8ms for the 2^5 division ratio.



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 zero forcing the microcontroller to begin program execution from the lowest Program Memory address.

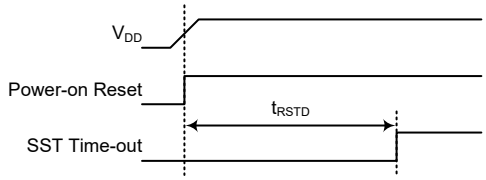
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

Reset Functions

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

Power-on Reset

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



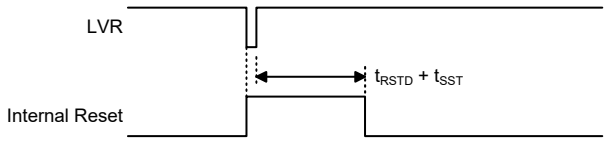
Power-on Reset Timing Chart

Low Voltage Reset – LVR

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

The LVR function is always enabled in normal operation with a specific LVR voltage V_{LVR}. For the device the V_{LVR} value is fixed at 2.1V. 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 in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVD/LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual t_{LVR} value can be selected by the TLVR1~TLVR0 bits in the TLVRC register.

Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.



Low Voltage Reset Timing Chart

- **TLVRC Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **TLVR1~TLVR0**: Minimum low voltage width to reset time (t_{LVR})
00: (7~8)×t_{LIRC}
01: (31~32)×t_{LIRC}
10: (63~64)×t_{LIRC}
11: (127~128)×t_{LIRC}

- **RSTFC Register**

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

"x": unknown

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set to 1 when an actual Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

Bit 1 Unimplemented, read as "0"

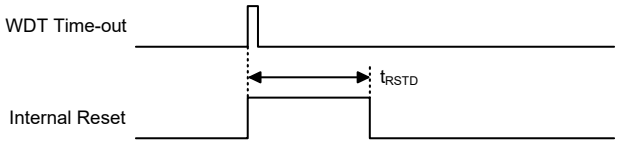
Bit 0 **WRF**: WDTC register software reset flag
Refer to the Watchdog Timer Control Register section.

IAP Reset

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

Watchdog Time-out Reset during Normal Operation

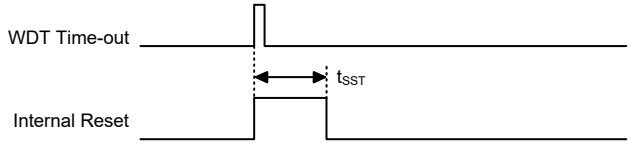
When the Watchdog Time-out Reset during normal operation in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

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

"u" 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	Cleared after reset, WDT begins counting
Timer Modules	Timer Modules will be turned off
Input/Output Ports	I/O ports will be setup as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers. Note that where more than one package type exists the table will reflect the situation for the larger package type.

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	--xx xxxx	--uu uuuu	--uu uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
PBP	---- ---0	---- ---0	---- ---u
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- -x-0	---- -u-u	---- -u-u
SCC	111- 0000	111- 0000	uuu- uuuu
HIRCC	---- 0000	---- 0000	---- uuuu
HXTC	---- -000	---- -000	---- -uuu
LXTC	---- -000	---- -000	---- -uuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
PC	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	uuuu uuuu
PD	-111 1111	-111 1111	-uuu uuuu
PDC	-111 1111	-111 1111	-uuu uuuu
PDPU	-000 0000	-000 0000	-uuu uuuu
WDTC	0101 0011	0101 0011	uuuu uuuu
SDSW0	1000 0000	1000 0000	uuuu uuuu
SDSW1	0-00 ---0	0-00 ---0	u-uu ---u
SDPGAC0	--00 0000	--00 0000	--uu uuuu
SDPGAC1	0000 0000	0000 0000	uuuu uuuu
SDA0C	-00- --00	-00- --00	-uu- --uu
SDA0VOS	0010 0000	0010 0000	uuuu uuuu
SDA1C	-00- --00	-00- --00	-uu- --uu
SDA1VOS	0010 0000	0010 0000	uuuu uuuu
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	uuuu uuuu
SADC2	0--- --10	0--- --10	u--- --uu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
PLTSW	---- 0001	---- 0001	---- uuuu
PLTDACC	---0 0000	---0 0000	---u uuuu
PLTDA0L	--00 0000	--00 0000	--uu uuuu

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PLTDA1L	--00 0000	--00 0000	--uu uuuu
PLTDA2L	--00 0000	--00 0000	--uu uuuu
PLTC0C	0000 0000	0000 0000	uuuu uuuu
PLTC0VOS	-001 0000	-001 0000	-uuu uuuu
PLTC1C	0000 0000	0000 0000	uuuu uuuu
PLTC1VOS	-001 0000	-001 0000	-uuu uuuu
PLTCHYC	-000 0000	-000 0000	-uuu uuuu
PLTAC	-00- ---0	-00- ---0	-uu- ---u
PLTAVOS	0010 0000	0010 0000	uuuu uuuu
PLTDICC1	000- -000	000- -000	uuu- -uuu
PLTDICC0	0--- --00	0--- --00	u--- --uu
INTEG	---- 0000	---- 0000	---- uuuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	-000 -000	-000 -000	-uuu -uuu
MFI0	--00 --00	--00 --00	--uu --uu
MFI1	--00 --00	--00 --00	--uu --uu
MFI2	--00 --00	--00 --00	--uu --uu
MFI3	-000 -000	-000 -000	-uuu -uuu
MFI4	-000 -000	-000 -000	-uuu -uuu
MFI5	--00 --00	--00 --00	--uu --uu
PSCR	---- -000	---- -000	---- -uuu
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
SIMC0	111- 0000	111- 0000	uuu- uuuu
SIMC1	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2	0000 0000	0000 0000	uuuu uuuu
SIMTOC	0000 0000	0000 0000	uuuu uuuu
U0SR	0000 1011	0000 1011	uuuu uuuu
U0CR1	0000 00x0	0000 00x0	uuuu uuuu
U0CR2	0000 0000	0000 0000	uuuu uuuu
U0CR3	---- ---0	---- ---0	---- ---u
BRDH0	0000 0000	0000 0000	uuuu uuuu
BRDL0	0000 0000	0000 0000	uuuu uuuu
UFCR0	--00 0000	--00 0000	--uu uuuu
TXR_RXR0	xxxx xxxx	xxxx xxxx	uuuu uuuu
RxCNT0	---- -000	---- -000	---- -uuu
U1SR	0000 1011	0000 1011	uuuu uuuu
U1CR1	0000 00x0	0000 00x0	uuuu uuuu
U1CR2	0000 0000	0000 0000	uuuu uuuu
U1CR3	---- ---0	---- ---0	---- ---u
BRDH1	0000 0000	0000 0000	uuuu uuuu
BRDL1	0000 0000	0000 0000	uuuu uuuu
UFCR1	--00 0000	--00 0000	--uu uuuu
TXR_RXR1	xxxx xxxx	xxxx xxxx	uuuu uuuu

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
RxCNT1	---- -000	---- -000	---- -uuu
STM0C0	0000 0000	0000 0000	uuuu uuuu
STM0C1	0000 0000	0000 0000	uuuu uuuu
STM0DL	0000 0000	0000 0000	uuuu uuuu
STM0DH	---- --00	---- --00	---- --uu
STM0AL	0000 0000	0000 0000	uuuu uuuu
STM0AH	---- --00	---- --00	---- --uu
PTM0C0	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	uuuu uuuu
PTM0C2	---- -000	---- -000	---- -uuu
PTM0DL	0000 0000	0000 0000	uuuu uuuu
PTM0DH	---- --00	---- --00	---- --uu
PTM0AL	0000 0000	0000 0000	uuuu uuuu
PTM0AH	---- --00	---- --00	---- --uu
PTM0BL	0000 0000	0000 0000	uuuu uuuu
PTM0BH	---- --00	---- --00	---- --uu
PTM0RPL	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	---- --00	---- --00	---- --uu
DAH	0000 0000	0000 0000	uuuu uuuu
DAL	0000 0000	0000 0000	uuuu uuuu
DACC	---- ---0	---- ---0	---- ---u
FC0	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	uuuu uuuu
FC2	---- ---0	---- ---0	---- ---u
FARL	0000 0000	0000 0000	uuuu uuuu
FARH	--00 0000	--00 0000	--uu uuuu
FD0L	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	uuuu uuuu
VBGRC	---- ---0	---- ---0	---- ---u
ORMC	0000 0000	0000 0000	0000 0000
LVDC	0000 0000	0000 0000	uuuu uuuu
TLVRC	---- --01	---- --01	---- --uu
CRCCR	---- ---0	---- ---0	---- ---u
CRCIN	0000 0000	0000 0000	uuuu uuuu
CRCDL	0000 0000	0000 0000	uuuu uuuu
CRCDH	0000 0000	0000 0000	uuuu uuuu
IECC	0000 0000	0000 0000	uuuu uuuu
STM1C0	0000 0000	0000 0000	uuuu uuuu
STM1C1	0000 0000	0000 0000	uuuu uuuu
STM1DL	0000 0000	0000 0000	uuuu uuuu
STM1DH	---- --00	---- --00	---- --uu

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
STM1AL	0000 0000	0000 0000	uuuu uuuu
STM1AH	---- --00	---- --00	---- --uu
PTM1C0	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	uuuu uuuu
PTM1DL	0000 0000	0000 0000	uuuu uuuu
PTM1DH	---- --00	---- --00	---- --uu
PTM1AL	0000 0000	0000 0000	uuuu uuuu
PTM1AH	---- --00	---- --00	---- --uu
PTM1RPL	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	---- --00	---- --00	---- --uu
PTM2C0	0000 0---	0000 0---	uuuu u---
PTM2C1	0000 0000	0000 0000	uuuu uuuu
PTM2DL	0000 0000	0000 0000	uuuu uuuu
PTM2DH	---- --00	---- --00	---- --uu
PTM2AL	0000 0000	0000 0000	uuuu uuuu
PTM2AH	---- --00	---- --00	---- --uu
PTM2RPL	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	---- --00	---- --00	---- --uu
SLEDC0	0000 0000	0000 0000	uuuu uuuu
SLEDC1	0000 0000	0000 0000	uuuu uuuu
IOSMTC	0101 0101	0101 0101	uuuu uuuu
PMPS0	0000 0000	0000 0000	uuuu uuuu
PMPS1	--00 0000	--00 0000	--uu uuuu
PCRL	0000 0000	0000 0000	uuuu uuuu
PCRH	--00 0000	--00 0000	--uu uuuu
STKPTR	0--- 0000	0--- 0000	u--- 0000
EEA	0000 0000	0000 0000	uuuu uuuu
EED	0000 0000	0000 0000	uuuu uuuu
EEC	---- 0000	---- 0000	---- uuuu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 0000	0000 0000	uuuu uuuu
PBS1	0000 0000	0000 0000	uuuu uuuu
PCS0	0000 0000	0000 0000	uuuu uuuu
PCS1	0000 0000	0000 0000	uuuu uuuu
PDS0	0000 0000	0000 0000	uuuu uuuu
PDS1	--00 0000	--00 0000	--uu uuuu
IFS0	0000 0000	0000 0000	uuuu uuuu
IFS1	--00 0000	--00 0000	--uu uuuu
ISGENC	0--- --00	0--- --00	u--- --uu
ISGDATA0	---0 0000	---0 0000	---u uuuu
ISGDATA1	---0 0000	---0 0000	---u uuuu

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

Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	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	—	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	—	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	—	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0

"—": Unimplemented, read as "0"

I/O Logic Function Register List

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using 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	PxPU6	PxPU5	PxPU4	PxPU3	PxPU2	PxPU1	PxPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

PxPUn: I/O Port x Pin pull-high function control
0: Disable
1: Enable

The PxPUn bit is used to control the pin pull-high function. Here the "x" can be A, B, C 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 register only when the pin is selected as a general purpose input and the MCU enters the IDLE or SLEEP mode.

• **PAWU Register**

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

Bit 7~0 **PAWU7~PAWU0**: PA7~PA0 wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

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

• **PxC Register**

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

PxCn: I/O Port x Pin type selection
0: Output
1: Input

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

The WDI and ENCLK are pin-shard with PA3 and PA6 pins respectively, the PAC3 and PAC6 bits should be properly configured to ensure correct signal direction.

I/O Port Schmitt Trigger Control

This function aims to reduce the power-on current before using the I/O functions including the pin-shared functions. It is only available when the pins are set to the GPIO functions by the pin-shared function selection of the PxSn register. If the pins are required to be set to other pin-shared functions, users should first set the IOSMTC register to any value other than 55H. If the pins support

the READ PORT function and the IECC register value has been set to 0CAH, the Schmitt Trigger function will be always enabled and the GPIO ports are not controlled by the IOSMTC register. However, the GPIO ports can be controlled by the IOSMTC register when the IECC register value is not equal to 0CAH.

• **IOSMTC Register**

Bit	7	6	5	4	3	2	1	0
Name	IOSMT7	IOSMT6	IOSMT5	IOSMT4	IOSMT3	IOSMT2	IOSMT1	IOSMT0
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 **IOSMT7~ IOSMT0**: GPIO port schmitt trigger control
01010101: Input gating
Others: Release input gating

These are the GPIO port schmitt trigger control bits. These bits will be automatically reset to 01010101b by hardware after any reset (except the WDT time-out reset in HALT state) to reduce the power-on current and when set to other values (not 01010101b) by the application program will release the GPIO input gating status. Before setting these bits to other values to release input gating, the application program must first set GPIO ports as input mode with pull-up resistor or as output mode to avoid additional power consumption.

Here the GPIO ports do not include PA0 and PA2, which are pin-shared with ICPDA/ICPCK/OCDSDA/OCDSCK. User should use an external circuit to solve the extra power consumption during the power-on period caused by the above-mentioned ICPDA/ICPCK/OCDSDA/OCDSCK pins floating.

I/O Port Power Source Control

This device supports different I/O port power source selections for PA0, PA2, PA3, PA6, PA7, PC4 and PC5 pins. With the exception of $\overline{\text{RES}}$/OCDS, the multi-power function is only effective when the pin is set to have a digital input or output function.

The port power can come from the power pin VDD or VDDIO, which is determined using the corresponding bit fields in the PMPS0~PMPS1 registers. The VDDIO power pin function should first be selected using the corresponding pin-shared function selection bits if the port power is supposed to come from the VDDIO pin.

An important point to know is that the input power voltage on the VDDIO pin should be equal to or less than the device supply power voltage V_{DD} when the VDDIO pin is selected as the port power supply pin.

• **PMPS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PMPS07	PMPS06	PMPS05	PMPS04	PMPS03	PMPS02	PMPS01	PMPS00
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 **PMPS07~PMPS06**: PA6 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}

If the PB4 pin is switched to the VDDIO function, and the PMPS07~PMPS06 bits are set to “1x”, the VDDIO pin input voltage can be used for PA6 pin power.

Bit 5~4 **PMPS05~PMPS04**: PA3 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}

If the PB4 pin is switched to the VDDIO function, and the PMPS05~PMPS04 bits are set to “1x”, the VDDIO pin input voltage can be used for PA3 pin power.

Bit 3~2 **PMPS03~PMPS02**: PA2 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}
If the PB4 pin is switched to the VDDIO function, and the PMPS03~PMPS02 bits are set to "1x", the VDDIO pin input voltage can be used for PA2 pin power.

Bit 1~0 **PMPS01~PMPS00**: PA0 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}
If the PB4 pin is switched to the VDDIO function, and the PMPS01~PMPS00 bits are set to "1x", the VDDIO pin input voltage can be used for PA0 pin power.

• **PMPS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PMPS15	PMPS14	PMPS13	PMPS12	PMPS11	PMPS10
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 **PMPS15~PMPS14**: PC5 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}
If the PB4 pin is switched to the VDDIO function, and the PMPS15~PMPS14 bits are set to "1x", the VDDIO pin input voltage can be used for PC5 pin power.

Bit 3~2 **PMPS13~PMPS12**: PC4 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}
If the PB4 pin is switched to the VDDIO function, and the PMPS13~PMPS12 bits are set to "1x", the VDDIO pin input voltage can be used for PC4 pin power.

Bit 1~0 **PMPS11~PMPS10**: PA7 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}
If the PB4 pin is switched to the VDDIO function, and the PMPS11~PMPS10 bits are set to "1x", the VDDIO pin input voltage can be used for PA7 pin power.

I/O Port Source Current Control

Each pin in this device can be configured with different output source current which is selected by the corresponding source current selection bits. These source current selection bits are available only when the corresponding pin is configured as a CMOS output. Otherwise, these select bits have no effect. Users should refer to the Input/Output Characteristics section to obtain the exact value for different applications.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLEDC0	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
SLEDC1	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10

Source Current Selection Register List

• **SLEDC0 Register**

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

Bit 7~6 **SLEDC07~SLEDC06**: 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	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **SLEDC17~SLEDC16**: PD6~PD4 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 5~4 **SLEDC15~SLEDC14**: PD3~PD0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 3~2 **SLEDC13~SLEDC12**: PC7~PC4 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 1~0 **SLEDC11~SLEDC10**: PC3~PC0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as 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 set as an input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
PDS1	—	—	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
IFS0	IFS07	IFS06	IFS05	IFS04	IFS03	IFS02	IFS01	IFS00
IFS1	—	—	IFS15	IFS14	IFS13	IFS12	IFS11	IFS10

Pin-shared Function Selection Register List

- **PAS0 Register**

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

Bit 7~6 **PAS07~PAS06**: PA3 Pin-shared function selection
00: PA3/INT0/STP1I
01: SDO

10: TX0
11: AN3

These bits should be set to 00 when it is required to use the WDI pin function of the Buzzer Driver.

Bit 5~4 **PAS05~PAS04**: PA2 Pin-shared function selection
00: PA2
01: SDI/SDA
10: RX0/TX0
11: PTP2B

Bit 3~2 **PAS03~PAS02**: PA1 Pin-shared function selection
00: PA1/INT1
01: $\overline{SCS}$
10: A1O
11: A1PI

Bit 1~0 **PAS01~PAS00**: PA0 Pin-shared function selection
00: PA0
01: SCK/SCL
10: PTP1B
11: PA0

• **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/STP0I/PTP0I
01: SCK/SCL
10: AN1
11: VREF

Bit 5~4 **PAS15~PAS14**: PA6 Pin-shared function selection
00: PA6
01: PTP0
10: SDI/SDA
11: RX0/TX0

These bits should be set to 01 when it is required to use the ENCLK pin function of the Buzzer Driver.

Bit 3~2 **PAS13~PAS12**: PA5 Pin-shared function selection
00: PA5/STCK0
01: STP1B
10: A1O
11: PA5/STCK0

Bit 1~0 **PAS11~PAS10**: PA4 Pin-shared function selection
00: PA4/PTCK0
01: STP0B
10: AN0
11: A0O

• **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: PLRX
10: SDI/SDA
11: RX0/TX0

Bit 5~4 **PBS05~PBS04**: PB2 Pin-shared function selection
00: PB2
01: PLIS
10: SCK/SCL
11: XT2

Bit 3~2 **PBS03~PBS02**: PB1 Pin-shared function selection
00: PB1
01: PLTX
10: TX0
11: XT1

Bit 1~0 **PBS01~PBS00**: PB0 Pin-shared function selection
00: PB0/INT0
01: STP0
10: A0PB
11: A0PI1

• **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PBS17~PBS16**: PB7 Pin-shared function selection
00: PB7
01: $\overline{\text{SCS}}$
10: AN6
11: PTP1

Bit 5~4 **PBS15~PBS14**: PB6 Pin-shared function selection
00: PB6
01: DACO
10: PLDICO
11: PTP2

Bit 3~2 **PBS13~PBS12**: PB5 Pin-shared function selection
00: PB5
01: TX0
10: STP0B
11: PLVREF

Bit 1~0 **PBS11~PBS10**: PB4 Pin-shared function selection
00: PB4
01: AN2
10: VDDIO
11: DACO

• **PCS0 Register**

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

Bit 7~6 **PCS07~PCS06**: PC3 Pin-shared function selection
00: PC3
01: SCK/SCL
10: TX1
11: PC3

Bit 5~4 **PCS05~PCS04**: PC2 Pin-shared function selection
00: PC2
01: SDI/SDA
10: RX1/TX1
11: PC2

Bit 3~2 **PCS03~PCS02**: PC1 Pin-shared function selection
00: PC1
01: STP1
10: SDO
11: TX0

Bit 1~0 **PCS01~PCS00**: PC0 Pin-shared function selection
00: PC0
01: RX0/TX0
10: AN7
11: $\overline{SCS}$

• **PCS1 Register**

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

Bit 7~6 **PCS17~PCS16**: PC7 Pin-shared function selection
00: PC7
01: AN13
10: PC7
11: PC7

Bit 5~4 **PCS15~PCS14**: PC6 Pin-shared function selection
00: PC6
01: AN12
10: PC6
11: PC6

Bit 3~2 **PCS13~PCS12**: PC5 Pin-shared function selection
00: PC5
01: STP0
10: AN4
11: TX1

Bit 1~0 **PCS11~PCS10**: PC4 Pin-shared function selection
00: PC4/STCK1
01: PTP0B
10: RX1/TX1
11: AN5

• **PDS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PDS07~PDS06**: PD3 Pin-shared function selection
00: PD3/PTCK2
01: AN11
10: PTP1B
11: PD3/PTCK2

Bit 5~4 **PDS05~PDS04**: PD2 Pin-shared function selection
00: PD2/PTP2I
01: $\overline{\text{SCS}}$
10: AN10
11: PTP1

Bit 3~2 **PDS03~PDS02**: PD1 Pin-shared function selection
00: PD1/PTP1I
01: OSC1
10: AN8
11: PTP2

Bit 1~0 **PDS01~PDS00**: PD0 Pin-shared function selection
00: PD0/PTCK1
01: OSC2
10: AN9
11: PTP2B

• **PDS1 Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PDS15~PDS14**: PD6 Pin-shared function selection
00: PD6
01: PTP2
10: AN15
11: PD6

Bit 3~2 **PDS13~PDS12**: PD5 Pin-shared function selection
00: PD5
01: SDO
10: DACO
11: PD5

Bit 1~0 **PDS11~PDS10**: PD4 Pin-shared function selection
00: PD4
01: AN14
10: PD4
11: PD4

• **IFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	IFS07	IFS06	IFS05	IFS04	IFS03	IFS02	IFS01	IFS00
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 **IFS07~IFS06**: PTP0I input source selection
00: CXCAP
01: PA7
10: CXCAP
11: CXCAP
Note: The CXCAP signal is from the Power Line Transceiver comparator output signal.

Bit 5~4 **IFS05~IFS04**: $\overline{SCS}$ input source pin selection
00: PC0
01: PB7
10: PA1
11: PD2

Bit 3~2 **IFS03~IFS02**: SCK/SCL input source pin selection
00: PB2
01: PA0
10: PA7
11: PC3
Note: If the SPI Master Mode is selected, when the SIMEN bit is set high, the PB2, PA0, PA7 and PC3 pins all can be used as the SCK pin function ignoring the IFS0[3:2] bit settings.

Bit 1~0 **IFS01~IFS00**: SDI/SDA input source pin selection
00: PB3
01: PA2
10: PA6
11: PC2

• **IFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	IFS15	IFS14	IFS13	IFS12	IFS11	IFS10
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"

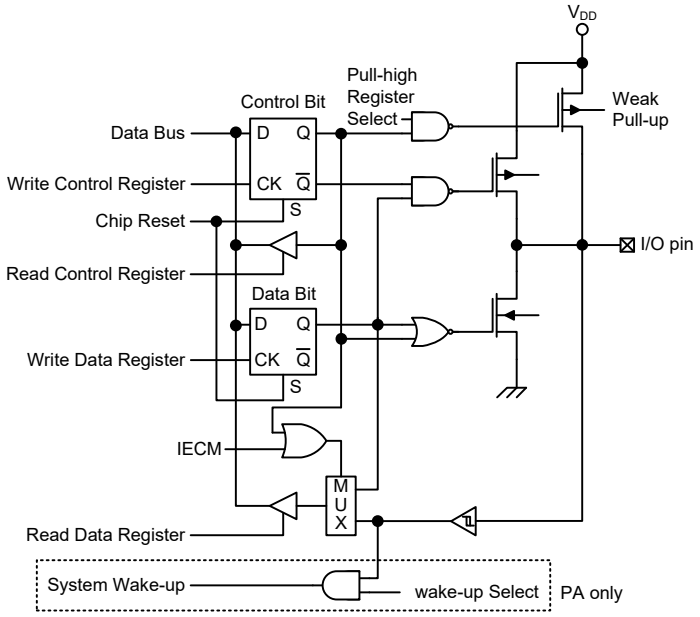
Bit 5~4 **IFS15~IFS14**: INT0 input source pin selection
00: PB0
01: PA3
10: PB0
11: PB0

Bit 3~2 **IFS13~IFS12**: RX1/TX1 input source pin selection
00: PC4
01: PC2
10: PC4
11: PC4

Bit 1~0 **IFS11~IFS10**: RX0/TX0 input source pin selection
00: PB3
01: PA2
10: PA6
11: PC0

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

READ PORT Function

The READ PORT function is used to manage the reading of the output data from the data latch or I/O pin, which is specially designed for the IEC 60730 self-diagnostic test on the I/O function and A/D paths. There is a register, IECC, which is used to control the READ PORT function. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function. When a specific data pattern, "11001010", is written into the IECC register, the internal signal named IECM will be set high to enable the READ PORT function. If the READ PORT function is enabled, the value on the corresponding pins will be passed to the accumulator ACC when the read port instruction "mov acc, Px" is executed where the "x" stands for the corresponding I/O port name.

Note that the READ PORT mode can only control the input path and will not affect the pin-shared function assignment and the current MCU operation. However, when the IECC register content is set to any other values rather than "11001010", the IECM internal signal will be cleared to 0 to disable the READ PORT function, and the reading path will be from the data latch. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function.

- **IECC Register**

Bit	7	6	5	4	3	2	1	0
Name	IECS7	IECS6	IECS5	IECS4	IECS3	IECS2	IECS1	IECS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

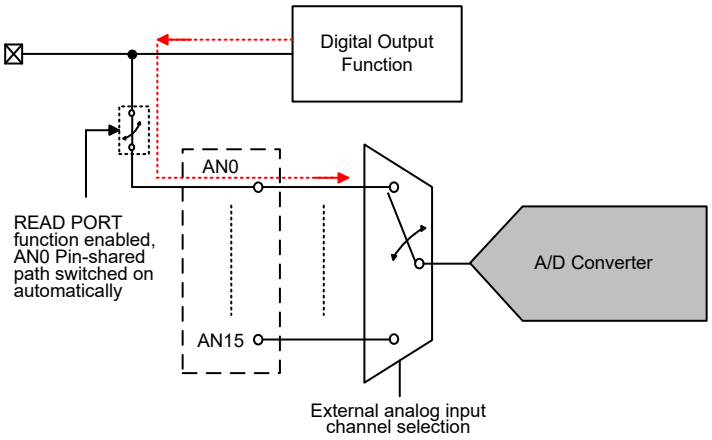
Bit 7~0 **IECS7~IECS0**: READ PORT function enable control bit 7 ~ bit 0
11001010: IECM=1 – READ PORT function is enabled
Others: IECM=0 – READ PORT function is disabled

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

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

The additional function of the READ PORT mode is to check the A/D path. When the READ PORT function is disabled, the A/D path from the external pin to the internal analog input will be switched off if the A/D input pin function is not selected by the corresponding selection bits. For the MCU with A/D converter channels, such as A/D AN15~AN0, the desired A/D channel can be switched on by properly configuring the external analog input channel selection bits in the A/D Control Register together with the corresponding analog input pin function is selected. However, the additional function of the READ PORT mode is to force the A/D path to be switched on. For example, when the AN0 is selected as the analog input channel as the READ PORT function is enabled, the AN0 analog input path will be switched on even if the AN0 analog input pin function is not selected. In this way, the AN0 analog input path can be examined by internally connecting the digital output on this shared pin with the AN0 analog input pin switch and then converting the corresponding digital data without any external analog input voltage connected.

Note that the A/D converter reference voltage should be equal to the I/O power supply voltage when examining the A/D path using the READ PORT function.



A/D Channel Input Path Internally Connection

Programming Considerations

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

instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

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

Timer Modules – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions the device includes several Timer Modules, 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 individual 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 Type TM sections.

Introduction

The device contains five TMs and each individual TM can be categorised as a certain type, namely Standard Type TM and 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 TMs will be described in this section. The detailed operation regarding each of the TM types will be described in separate sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

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

TM Function Summary

TM Operation

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

TM Clock Source

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

TM Interrupts

The Standard and Periodic type TMs each have two internal interrupts, one for each of the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs. 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 xTCKn and xTPnI respectively. The xTMn input pin, xTCKn, is essentially a clock source for the xTMn and is selected using the xTnCK2~xTnCK0 bits in the xTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The xTCKn input pin can be chosen to have either a rising or falling active edge. The xTCKn pin is also used as the external trigger input pin in single pulse output mode.

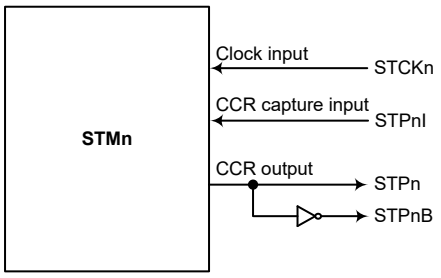
The other xTMn input pin, xTPnI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the xTnIO1~xTnIO0 bits in the xTMnC1 register. There is another capture input, PTCKn, for PTMn capture input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

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

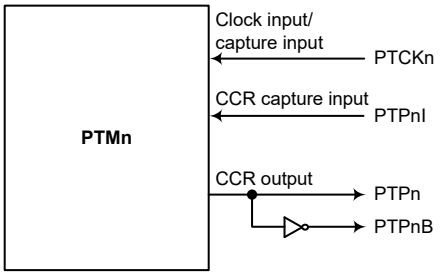
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using the relevant pin-shared function selection bits. The details of the pin-shared function selection are described in the pin-shared function section.

STMn		PTMn	
Input	**Output**	**Input**	**Output**
STCKn, STPnI	STPn, STPnB	PTCKn, PTPnI	PTPn, PTPnB

TM External Pins



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

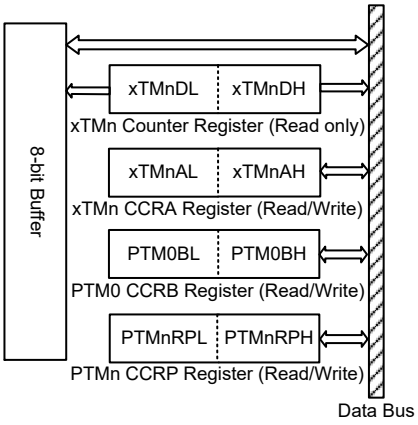


PTMn Function Pin Block Diagram (n=0~2)

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA and CCRP registers as well as the PTM0 CCRB register, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed.

As the CCRA, CCRB and CCRP registers are implemented in the way shown in the following diagram and accessing these register pairs is carried out in a specific way as described above, it is recommended to use the "MOV" instruction to access the CCRA, CCRB and CCRP low byte registers, named xTMnAL, PTM0BL, PTMnRPL, using the following access procedures. Accessing the CCRA, CCRB or CCRP low byte register without following these access procedures will result in unpredictable values.



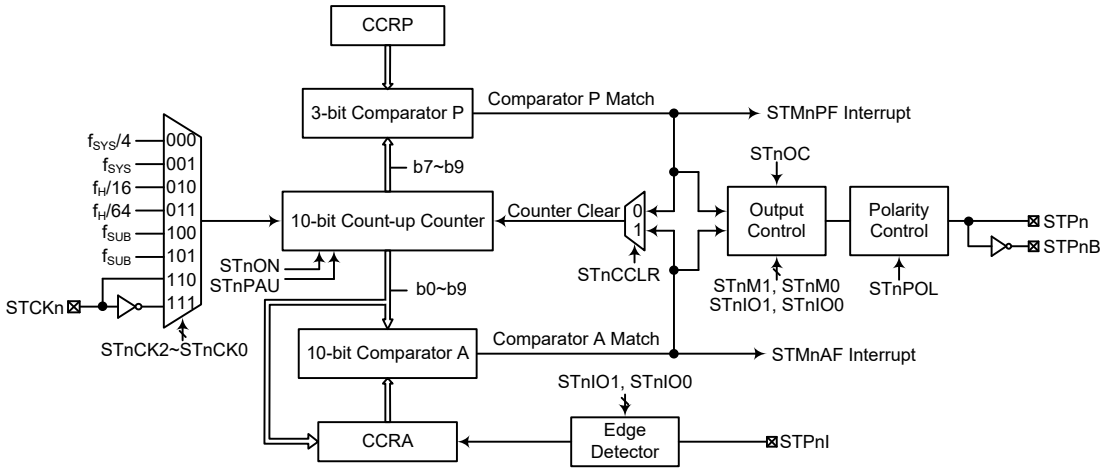
The following steps show the read and write procedures:

- Writing Data to CCRA, CCRB or CCRP
  - Step 1. Write data to Low Byte xTMnAL, PTM0BL or PTMnRPL
    - Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte xTMnAH, PTM0BH 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, CCRA, CCRB or CCRP
  - Step 1. Read data from the High Byte xTMnDH, xTMnAH, PTM0BH or PTMnRPH
    - Here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.

- Step 2. Read data from the Low Byte xTMnDL, xTMnAL, PTM0BL 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 type TM can also be controlled with two external input pins and can drive two external output pins.



Note: As the STMn external pins are pin-shared with other functions, so before using the STMn function, ensure that the relevant pin-shared function registers have been set properly to enable the STMn pin function. The STCKn and STPnI pin, if used, must also be set as an input by setting the corresponding bits in the port control register.

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

Standard Type TM Operation

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

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

Standard Type TM Register Description

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

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

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

- **STMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	STnPAU	STnCK2	STnCK1	STnCK0	STnON	STnRP2	STnRP1	STnRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **STnPAU**: STMn Counter Pause Control
0: Run
1: Pause

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

Bit 6~4 **STnCK2~STnCK0**: Select STMn Counter Clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: STCKn rising edge clock
111: STCKn falling edge clock

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

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

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

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

Bit 2~0 **STnRP2~STnRP0**: STMn CCRP 3-bit register, compared with the STMn counter bit 9~bit 7

Comparator P Match Period=
000: 1024 STMn clocks
001: 128 STMn clocks

010: 256 STMn clocks
011: 384 STMn clocks
100: 512 STMn clocks
101: 640 STMn clocks
110: 768 STMn clocks
111: 896 STMn clocks

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

• STMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	STnM1	STnM0	STnIO1	STnIO0	STnOC	STnPOL	STnDPX	STnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **STnM1~STnM0**: Select STMn Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

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

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

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

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

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the STnIO1 and STnIO0 bits determine how the STMn output pin changes state when a compare match occurs from the Comparator A. The STM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the STM output pin should be setup using the STnOC bit in the STMnC1 register. Note that the output

level requested by the STnIO1 and STnIO0 bits must be different from the initial value setup using the STnOC bit otherwise no change will occur on the STMn output pin when a compare match occurs. After the STMn output pin changes state, it can be reset to its initial level by changing the level of the STnON bit from low to high.

In the PWM Output Mode, the STnIO1 and STnIO0 bits determine how the STMn output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to change the values of the STnIO1 and STnIO0 bits only after the STMn has been switched off. Unpredictable PWM outputs will occur if the STnIO1 and STnIO0 bits are changed when the STMn is running.

Bit 3 **STnOC**: STPn Output control

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

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

This is the output control bit for the STMn output pin. Its operation depends upon whether STMn 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 STMn is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the STMn output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low. In the Single Pulse Output Mode it determines the logic level of the STPn output pin when the STnON bit changes from low to high.

Bit 2 **STPnOL**: STPn Output Polarity control
0: Non-inverted
1: Inverted

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

Bit 1 **STnDPX**: STMn PWM duty/period control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

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

Bit 0 **STnCCLR**: STMn Counter Clear condition selection
0: Comparator P match
1: Comparator A match

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

• **STMnDL Register**

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

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

• **STMnDH Register**

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

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

• **STMnAL Register**

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

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

• **STMnAH Register**

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

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

Standard Type TM Operating Modes

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

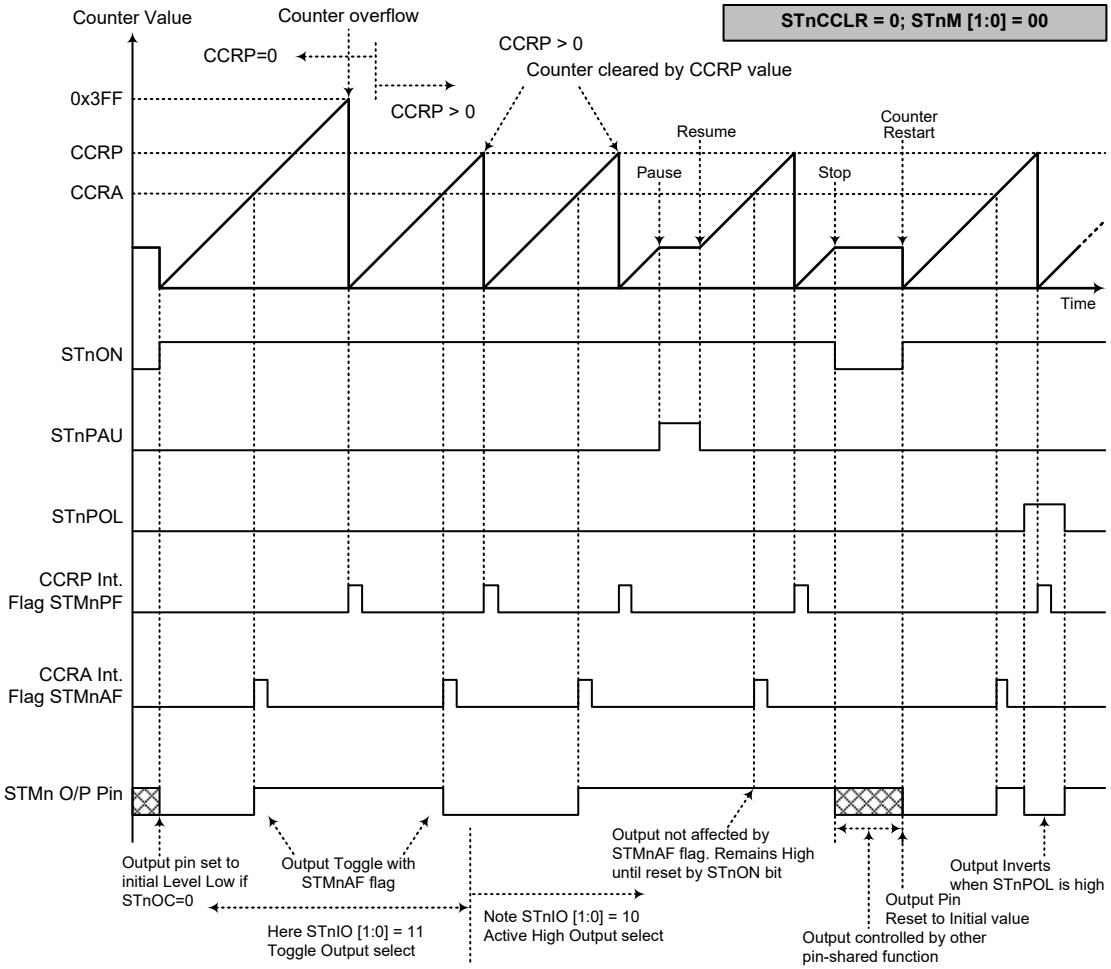
Compare Match Output Mode

To select this mode, bits STnM1 and STnM0 in the STMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the STnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both STMnAF and STMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the STnCCLR bit in the STMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when STnCCLR is high no STMnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be set to “0”.

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

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

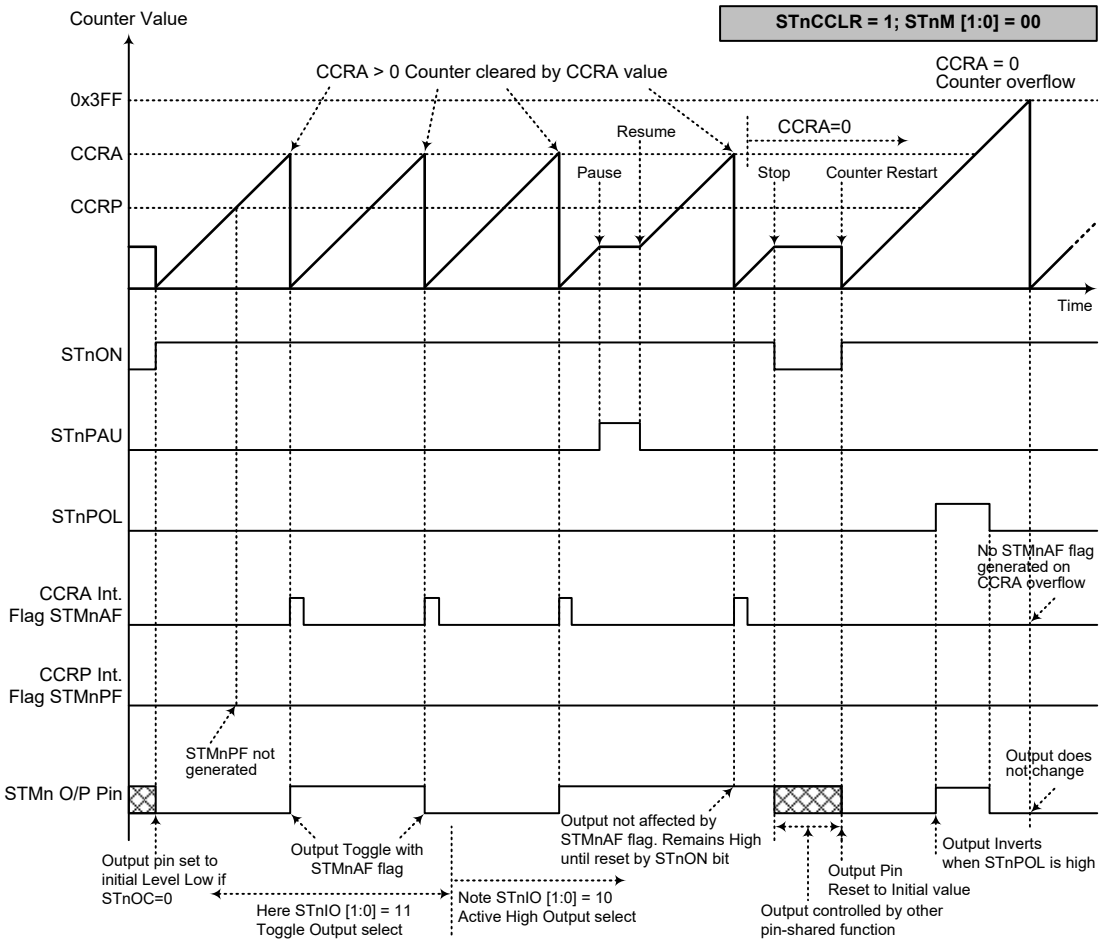


Compare Match Output Mode – STnCCLR=0 (n=0~1)

Note: 1. With STnCCLR=0, a Comparator P match will clear the counter

2. The STMn output pin controlled only by the STMnAF flag

3. The output pin reset to initial state by an STnON bit rising edge



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

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

2. The STMn output pin controlled only by the STMnAF flag

3. The output pin reset to initial state by an STnON rising edge

4. The STMnPF flags is not generated when STnCCLR=1

Timer/Counter Mode

To select this mode, bits STnM1 and STnM0 in the STMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the STMn output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the STMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits STnM1 and STnM0 in the STMnC1 register should be set to 10 respectively. The PWM function within the STMn is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the STMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output mode, the STnCCLR bit has no effect as the PWM period. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the STnDPX bit in the STMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The STnOC bit in the STMnC1 register is used to select the required polarity of the PWM waveform while the two STnIO1 and STnIO0 bits are used to enable the PWM output or to force the STMn output pin to a fixed high or low level. The STnPOL bit is used to reverse the polarity of the PWM output waveform.

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

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

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

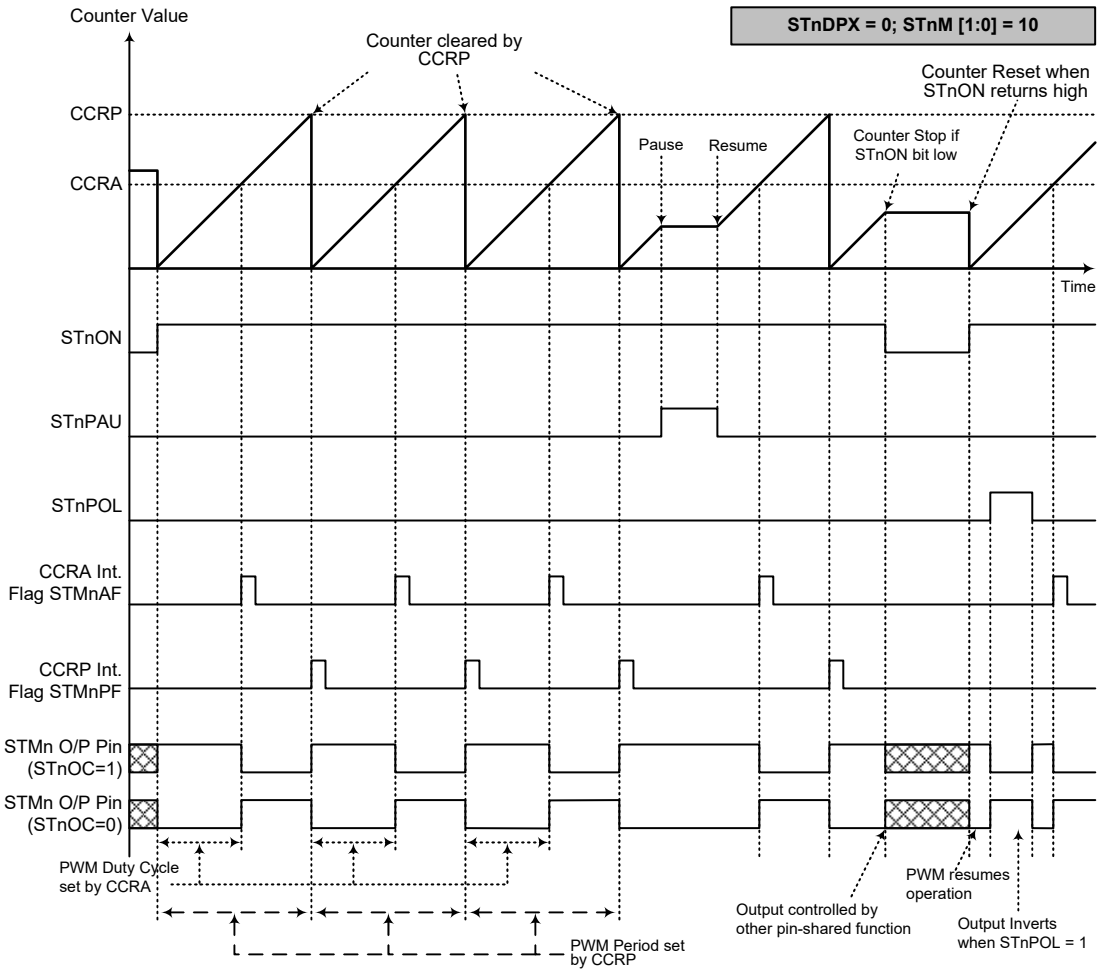
The STMn PWM output frequency=(f_{SYS}/4)/(4×128)=f_{SYS}/2048=8kHz, duty=128/(4×128)=25%.

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

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

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

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



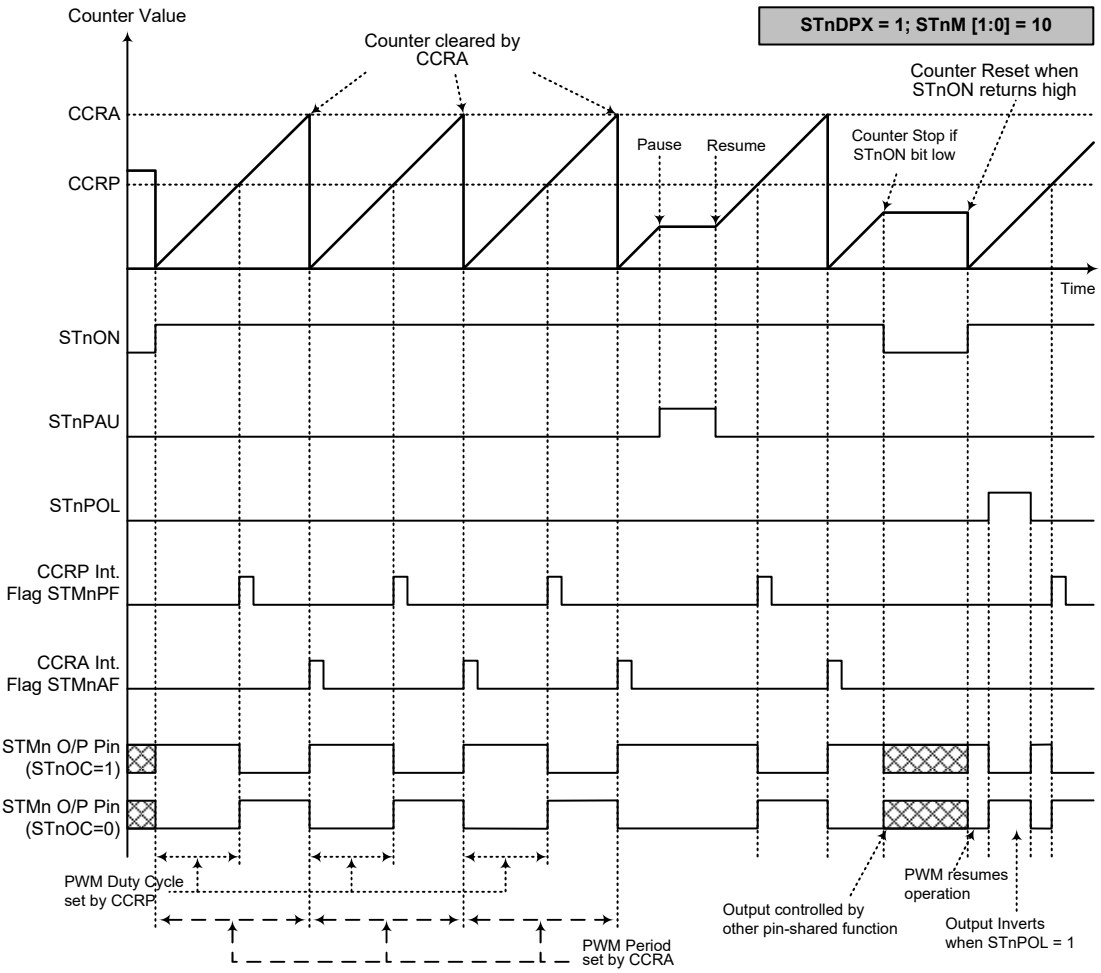
PWM Output Mode – STnDPX=0 (n=0~1)

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

2. A counter clear sets PWM Period

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

4. The STnCCLR bit has no influence on PWM operation



PWM Output Mode – STnDPX=1 (n=0~1)

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

2. A counter clear sets PWM Period

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

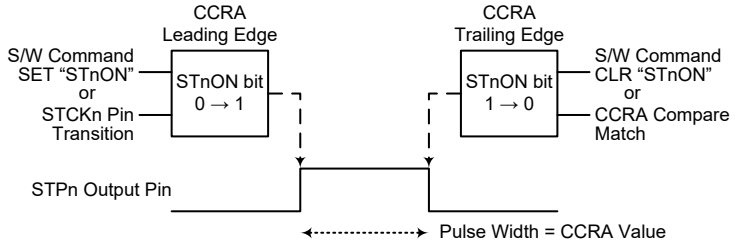
4. The STnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

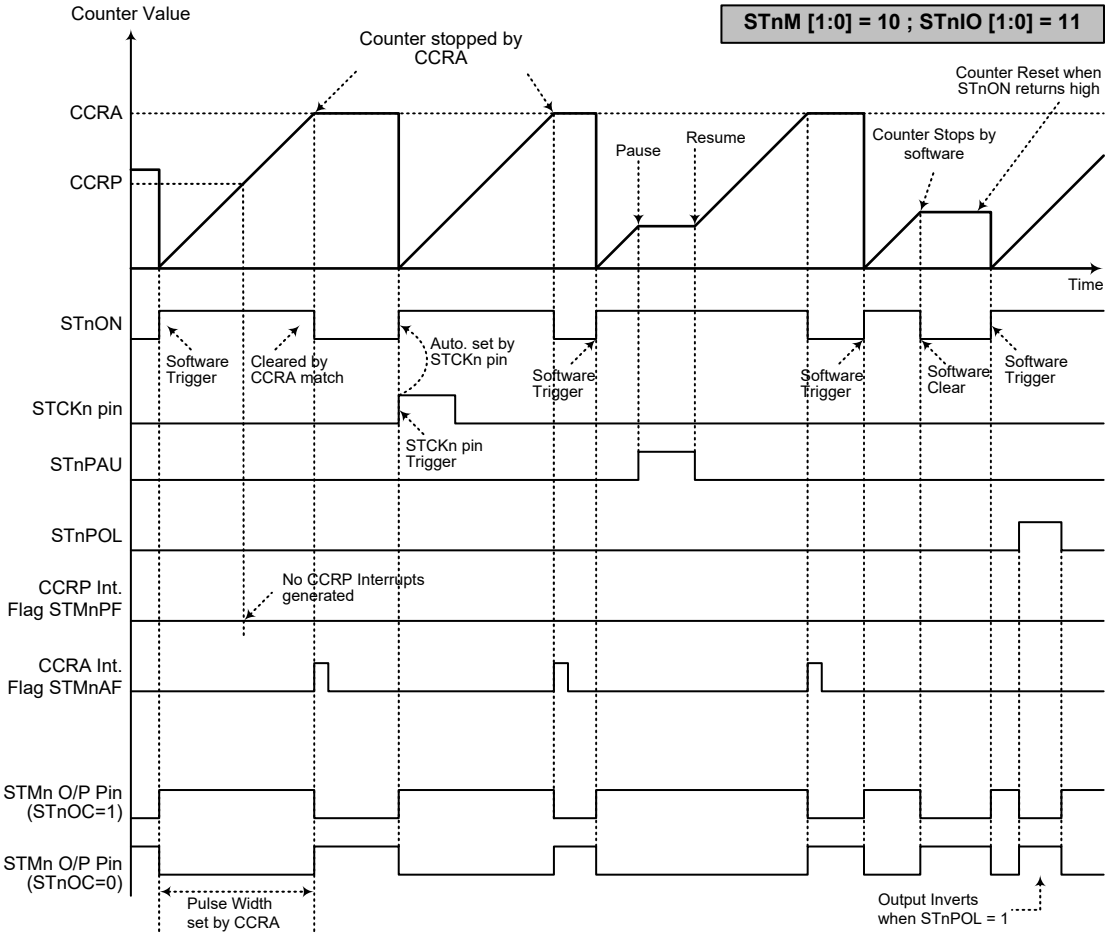
To select this mode, bits STnM1 and STnM0 in the STMnC1 register should be set to 10 respectively and also the STnIO1 and STnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STMn output pin.

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

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



Single Pulse Generation



Single Pulse Output Mode (n=0~1)

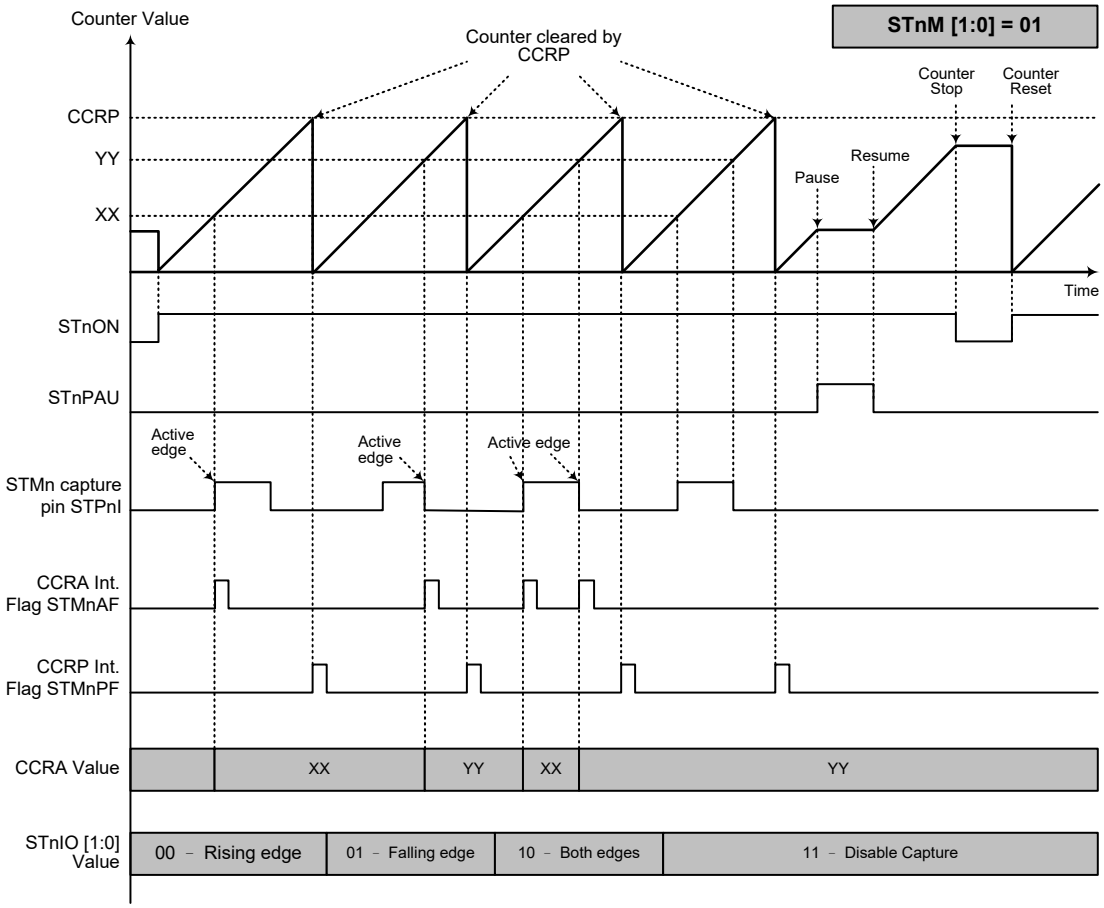
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse triggered by the STCKn pin or by setting the STnON bit high
4. An STCKn pin active edge will automatically set the STnON bit high
5. In the Single Pulse Output Mode, STnIO [1:0] must be set to “11” and cannot be changed

Capture Input Mode

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

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

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

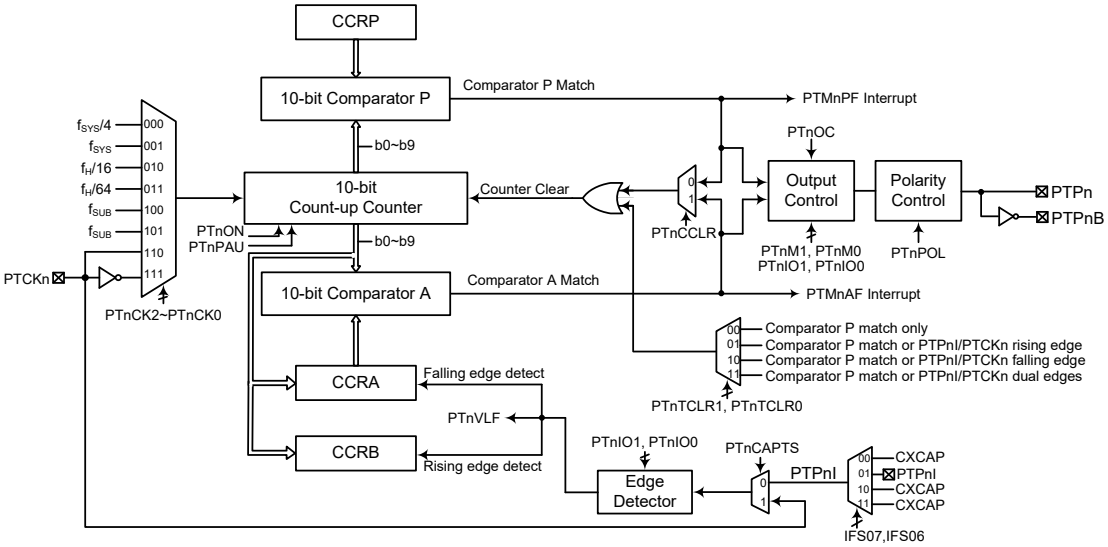


Capture Input Mode (n=0~1)

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

Periodic Type TM – PTM

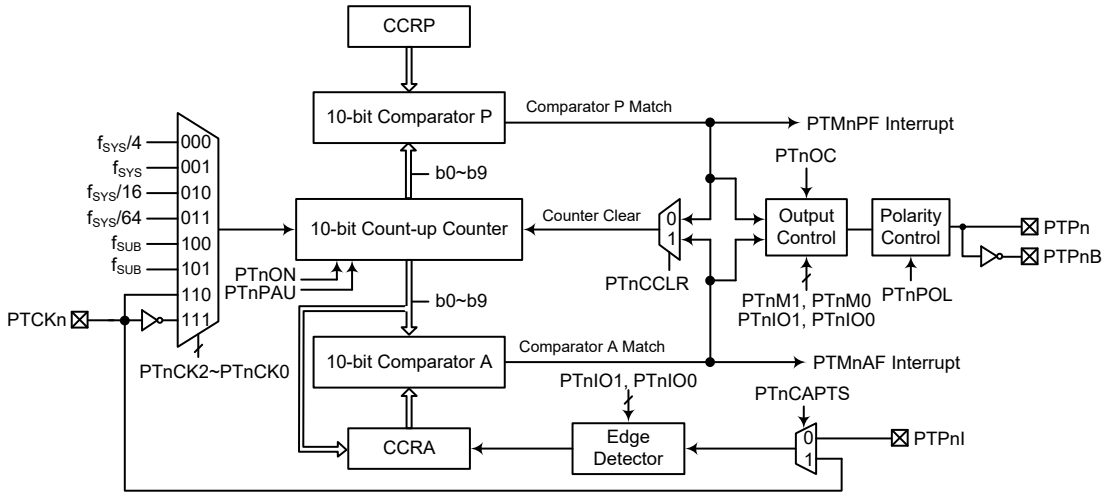
The Periodic Type TM contains five operating modes, which are Compare Match Output, Timer/ Event Counter, Capture Input, Single Pulse Output and PWM Output modes. The Periodic TMs can be controlled with two external input pins and can drive two external output pins.



Periodic Type TM Block Diagram (n=0)

Note: 1. The CXCAP is the PowerLine Transceiver comparator output signal.

2. As the PTMn external pins are pin-shared with other functions, so before using the PTMn function, ensure that the relevant pin-shared function registers have be set properly to enable the PTMn pin function. The PTCKn and PTPnI pins, if used, must also be set as an input by setting the corresponding bits in the port control register.



Note: If the PTMn external pins will be used and as these pins are pin-shared with other functions, before using the PTMn function, the pin-shared function registers should be set properly.

Periodic Type TM Block Diagram (n=1~2)

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 CCRA and CCRP

registers. The CCRA and CCRP comparators are 10-bit wide whose value is respectively compared with all counter bits.

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

Periodic Type TM Register Description

Overall operation of the Periodic Type TM is controlled using a series of registers. A read only register pair exists to store the internal 10-bit counter value, while three read/write register pairs exist to store the internal 10-bit CCRA value, CCRP value and CCRB value. The remaining three 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
PTMnC2*	—	—	—	—	—	PTnTCLR1	PTnTCLR0	PTnVLF
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnBL*	D7	D6	D5	D4	D3	D2	D1	D0
PTMnBH*	—	—	—	—	—	—	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

*: The registers with * symbol are only available for PTM0

10-bit Periodic TM Register List (n=0~2)

- **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**: PTMn counter clock selection
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}

101: f_{SUB}
110: PTCKn rising edge clock
111: PTCKn falling edge clock

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

Bit 3 **PTON**: 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**: PTMn operating mode selection
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

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

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

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

Capture Input Mode (n=0)

PTnTCLR[1:0]=00B:
00: Input capture at rising edge of PTPnI, CXCAP or PTCKn, and the counter value will be latched into CCRA
01: Input capture at falling edge of PTPnI, CXCAP or PTCKn, and the counter value will be latched into CCRA
10: Input capture at both falling and rising edges of PTPnI, CXCAP or PTCKn, and the counter value will be latched into CCRA
11: Input capture disabled

PTnTCLR[1:0]=01B, 10B or 11B:
- 00: Input capture at rising edge of PTPnI, CXCAP or PTCKn, and the counter value will be latched into CCRB
- 01: Input capture at falling edge of PTPnI, CXCAP or PTCKn, and the counter value will be latched into CCRA
- 10: Input capture at both falling and rising edges of PTPnI, CXCAP or PTCKn, and the counter value will be latched into CCRA at falling edge or CCRB at rising edge
- 11: Input capture disabled

Capture Input Mode (n=1~2)
- 00: Input capture at rising edge of PTPnI or PTCKn
- 01: Input capture at falling edge of PTPnI or PTCKn
- 10: Input capture at rising/falling edge of PTPnI or PTCKn
- 11: Input capture disabled

Timer/Counter Mode
- Unused

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

Bit 3 **PTnOC**: PTMn PTPn output control 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. Its operation depends upon whether PTMn is being used in the Compare Match Output Mode or in the PWM Output Mode/ Single Pulse Output Mode. It has no effect if the PTMn is in the Timer/Counter Mode. In the Compare Match Output Mode, it determines the logic level of the PTMn output before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low. In the Single Pulse Output Mode, it determines the logic level of the PTMn output when the PTnON bit changes from low to high.

Bit 2 **PTnPOL**: PTMn PTPn output polarity control
- 0: Non-invert
- 1: Invert

This bit controls the polarity of the PTPn output. When the bit is set high the PTMn output 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

n=0

0: From the external PTPnI pin or the internal CXCAP signal, selected using the IFS0[7:6] bits

1: From the PTCKn pin

n=1~2

0: From PTPnI pin

1: From PTCKn pin

Bit 0 **PTnCCLR**: PTMn counter clear condition selection

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 Mode or Capture Input Mode.

• PTMnC2 Register (n=0 only)

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PTnTCLR1	PTnTCLR0	PTnVLF
R/W	—	—	—	—	—	R/W	R/W	R
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2~1 **PTnTCLR1~PTnTCLR0**: PTMn counter clear condition selection (Capture Input Mode only)

00: Comparator P match

01: Comparator P match or PTCKn/PTPnI/CXCAP rising edge

10: Comparator P match or PTCKn/PTPnI/CXCAP falling edge

11: Comparator P match or PTCKn/PTPnI/CXCAP dual edges

Bit 0 **PTnVLF**: PTMn counter value latch edge flag

0: Falling edge triggers the counter value latch

1: Rising edge triggers the counter value latch

When the PTnTCLR1~PTnTCLR0 bits are equal to 00B, ignore this flag status.

• 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

• **PTMnBL Register(n=0 only)**

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 CCRB Low Byte Register bit 7 ~ bit 0
PTMn 10-bit CCRB bit 7 ~ bit 0

• **PTMnBH Register(n=0 only)**

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 CCRB High Byte Register bit 1 ~ bit 0
PTMn 10-bit CCRB 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.

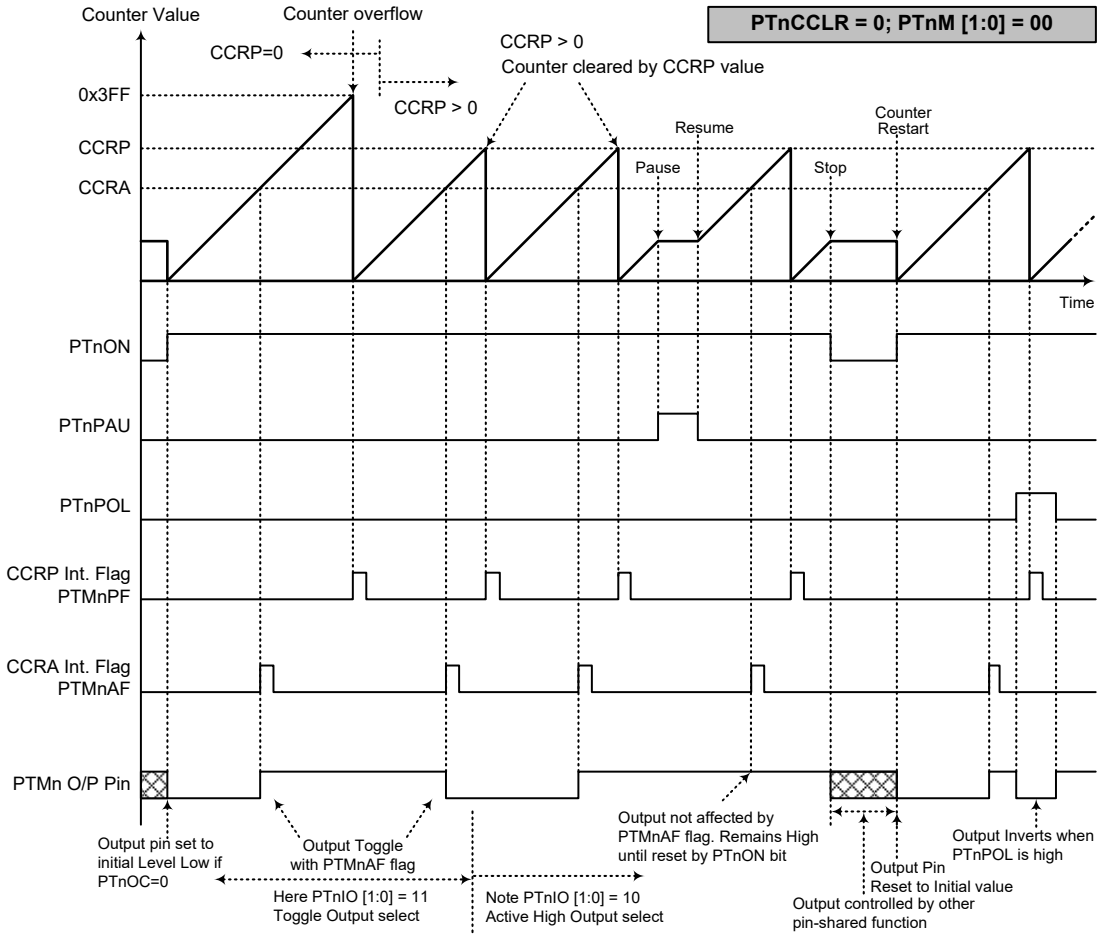
Compare Match Output Mode

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

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

If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 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 will change state. The PTMn output 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. The way in which the PTMn output changes state are determined by the condition of the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The PTMn output 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, 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 output change will take place.

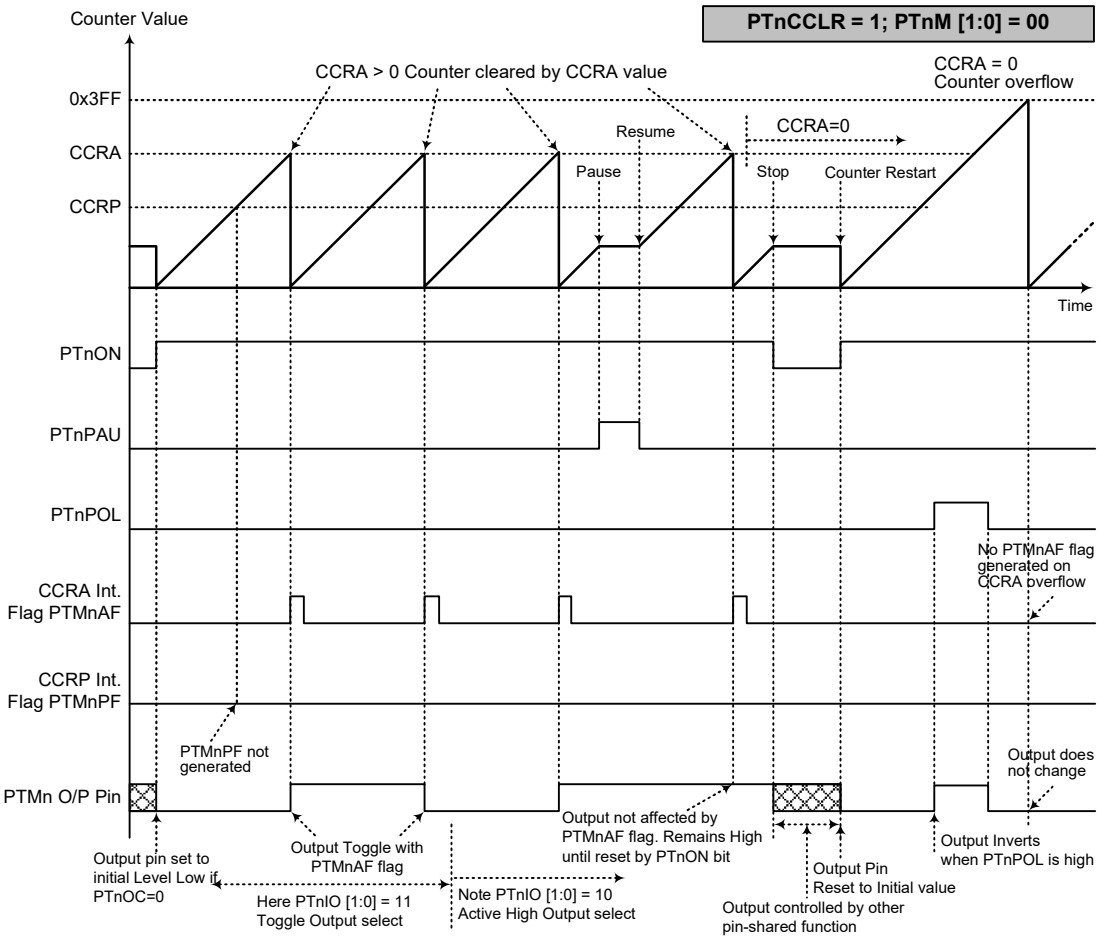


Compare Match Output Mode – PTnCCLR=0 (n=0~2)

Note: 1. With PTnCCLR=0 a Comparator P match will clear the counter

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

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



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

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

2. The PTMn output is controlled only by the PTMnAF flag
3. The output 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 pins are 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 pins are not used in this mode, the pins can be used as normal I/O pins or other pin-shared functions.

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, the CCRP register is used to clear the internal counter and thus control the PWM waveform frequency, while the CCRA register is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

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

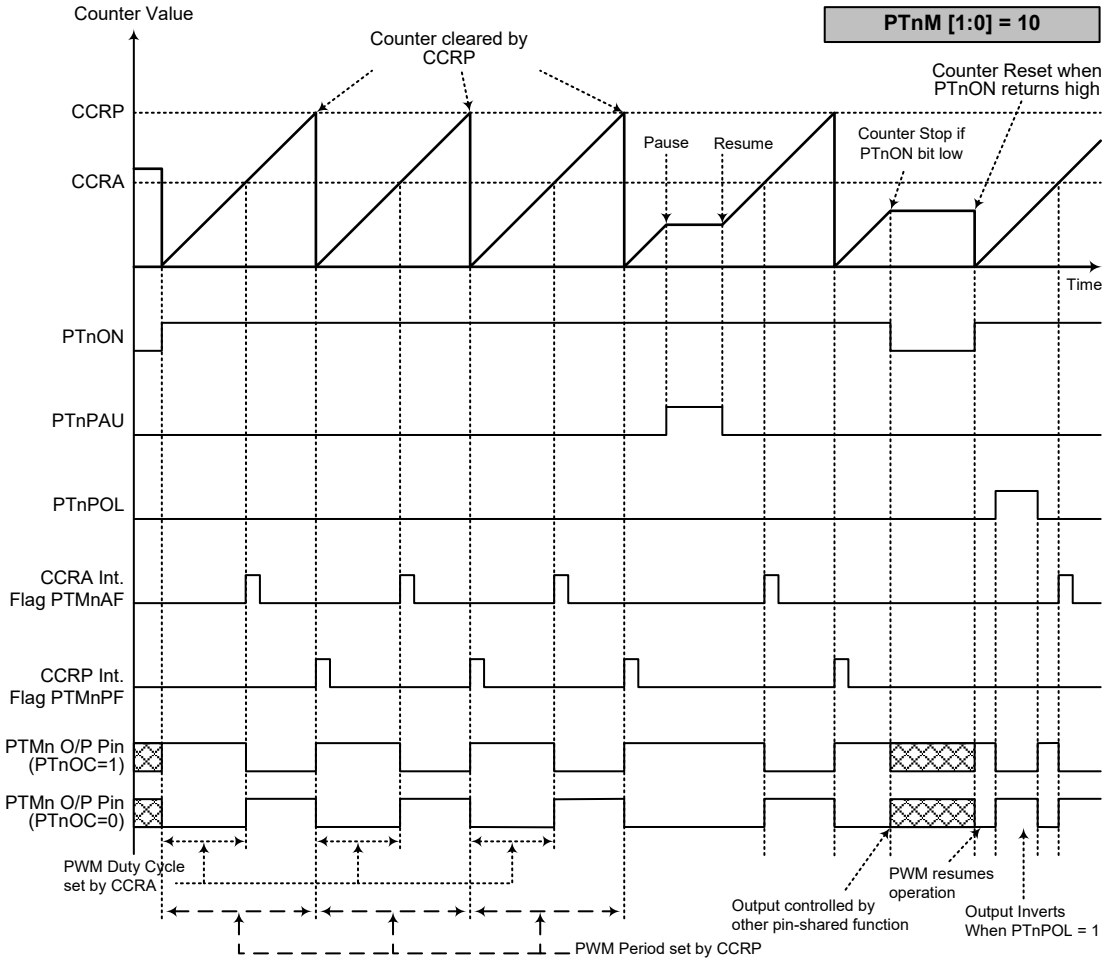
- **10-bit PTMn, PWM Output Mode, Edge-aligned Mode**

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

If f_{SYS}=16MHz, PTMn clock source select f_{SYS}/4, CCRP=512 and CCRA=128,

The PTMn PWM output frequency=(f_{SYS}/4)/512=f_{SYS}/2048=8kHz, 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=0~2)

Note: 1. The counter is cleared by CCRP

2. A counter clear sets the PWM Period

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

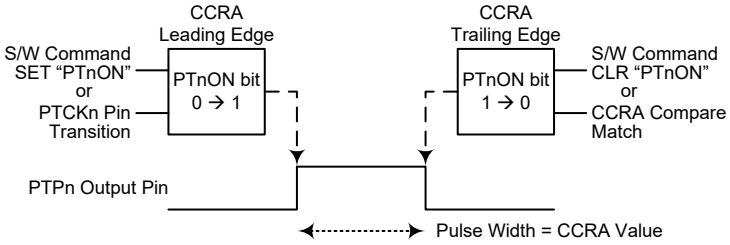
4. The PTnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

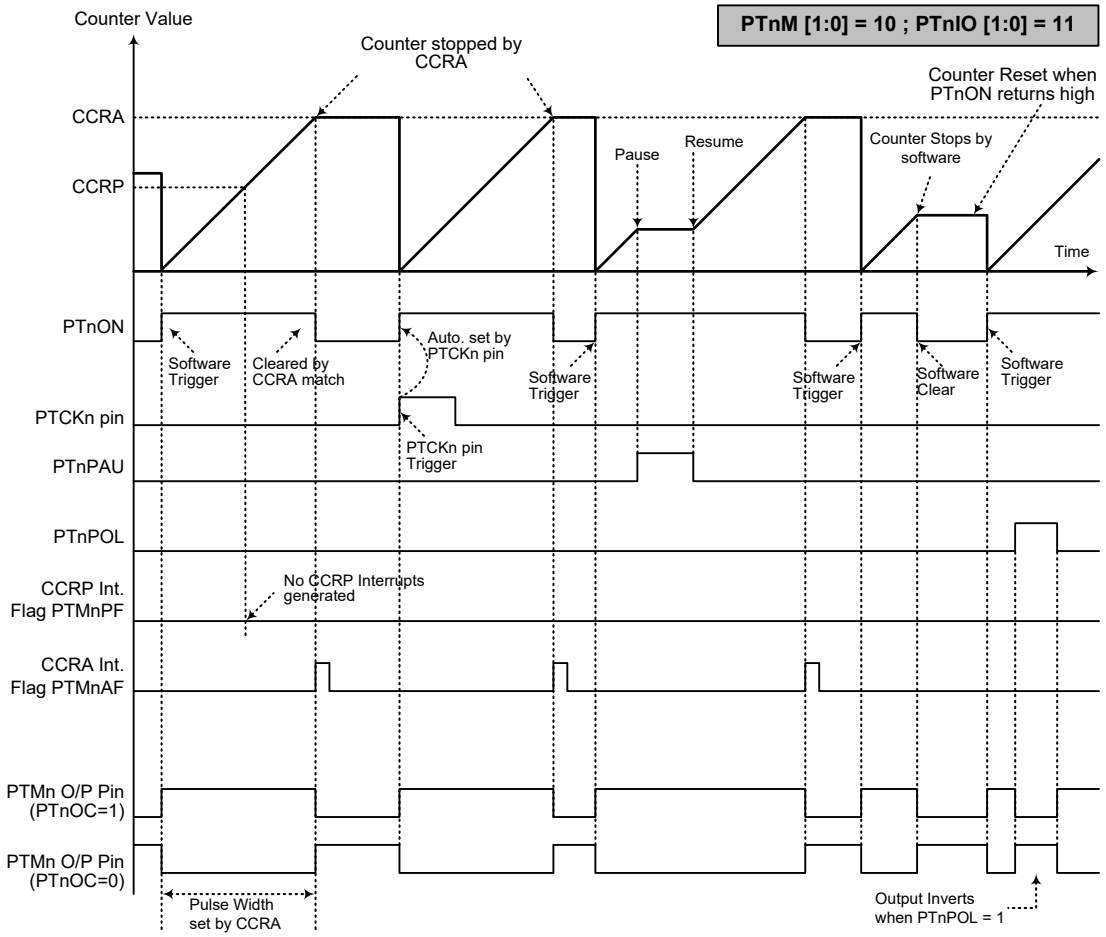
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

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

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



Single Pulse Generation (n=0~2)



Single Pulse Output Mode (n=0~2)

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 Mode, PTnIO[1:0] must be set to “11” and cannot be changed

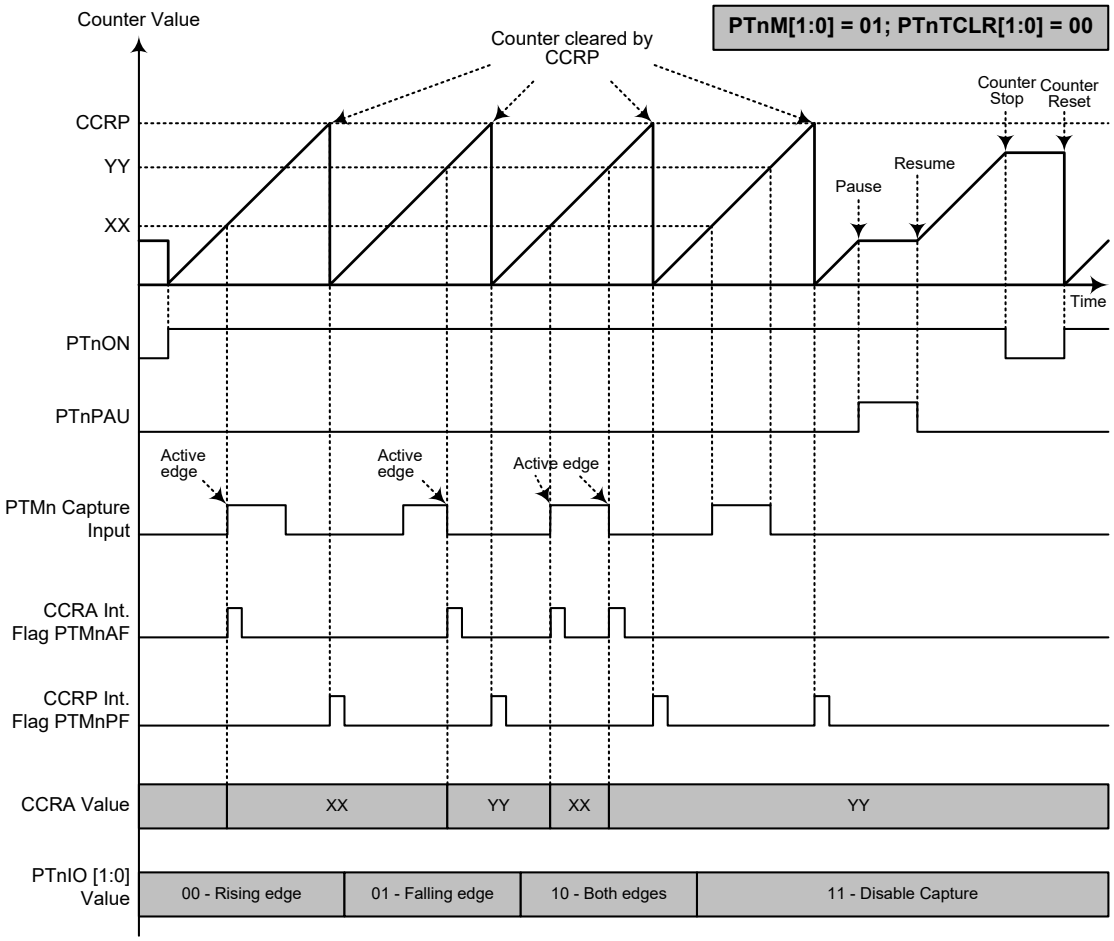
Capture Input Mode (n=0)

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 01 respectively. This mode enables external or internal 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 or internal signal is supplied on the PTCKn or PTPnI pin, which is selected using the PTnCAPTS bit in the PTMnC1 register. While the PTPnI signal can be from the external PTPnI pin input or from the internal CXCAP signal, which is selected using the IFS0[7:6] bits in the IFS0 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.

The PTnIO1 and PTnIO0 bits decide which active edge transition type to latch counter value and generate an interrupt. The PTnTCLR1 and PTnTCLR0 bits decide the condition that the counter reset back to zero. The present counter value being latched into CCRA or CCRB is decided by both PTnIO1~PTnIO0 and PTnTCLR1~PTnTCLR0 settings. The PTnIO1~PTnIO0 and PTnTCLR1~PTnTCLR0 settings are independent and uninfluenced on each other.

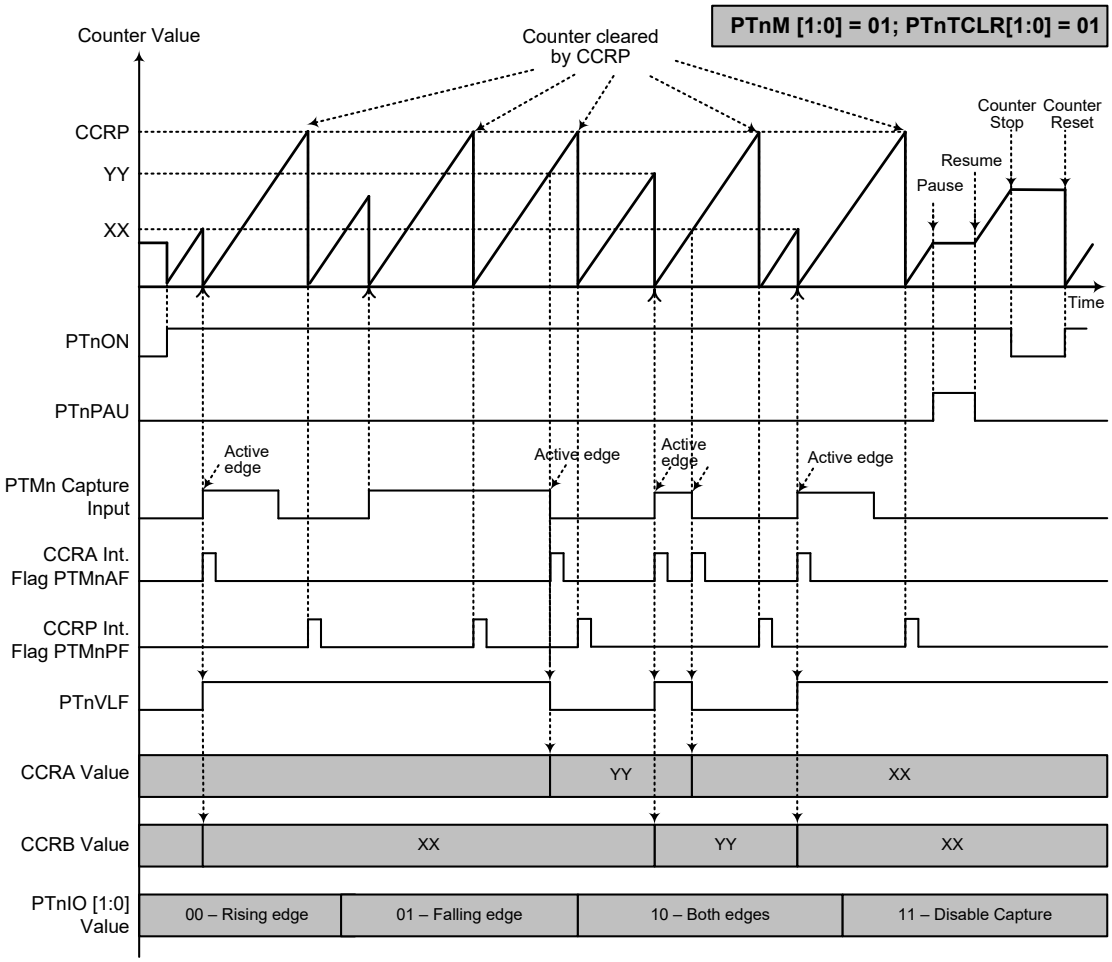
When the required edge transition appears on the PTCKn or PTPnI pin or CXCAP signal, the present value in the counter will be latched into the CCRA registers or CCRB registers and a PTMn interrupt will be generated. Irrespective of what events occur on the PTCKn or PTPnI pin or CXCAP signal, the counter will continue to free run until the PTnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTMn interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTnIO1 and PTnIO0 bits can select the active trigger edge on the PTCnK or PTPnI pin or CXCAP signal 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 PTCKn or PTPnI pin or CXCAP signal, however it must be noted that the counter will continue to run. The PTnCCLR, PTnOC and PTnPOL bits are not used in this mode.

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



Capture Input Mode – PTnTCLR[1:0]=00 (n=0)

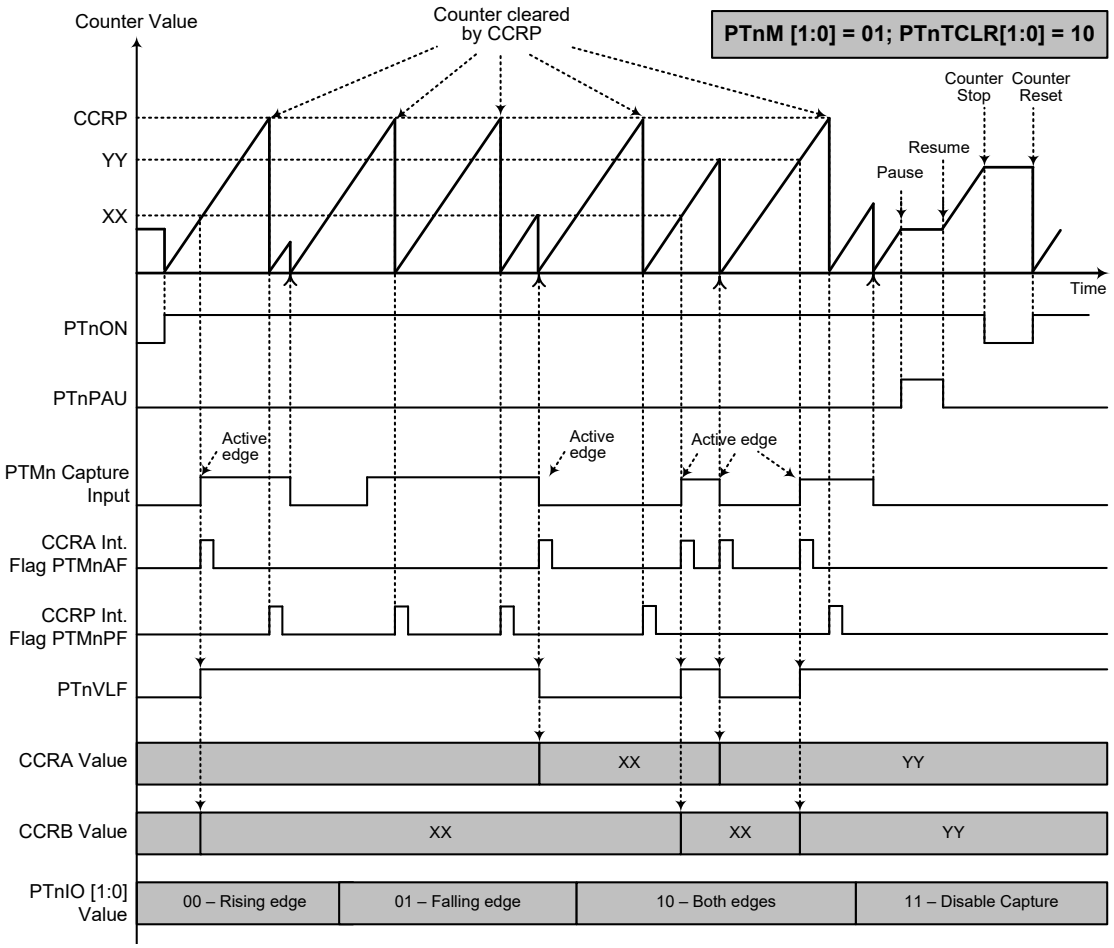
Note: 1. PTnM[1:0]=01, PTnTCLR[1:0]=00 and active edge set by the PTnIO[1:0] bits
2. A PTMn capture input (PTCKn, PTPnI pin or CXCAP signal) active edge transfers the counter value to CCRA
3. Comparator P match will clear the counter
4. PTnCCLR bit is not used
5. No output function – PTnOC and PTnPOL bits are not used
6. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero
7. Ignore the PTnVLF bit status when PTnTCLR[1:0]=00
8. The capture input mode cannot be used if the selected PTMn counter clock is not available



Capture Input Mode – PTnTCLR[1:0]=01 (n=0)

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

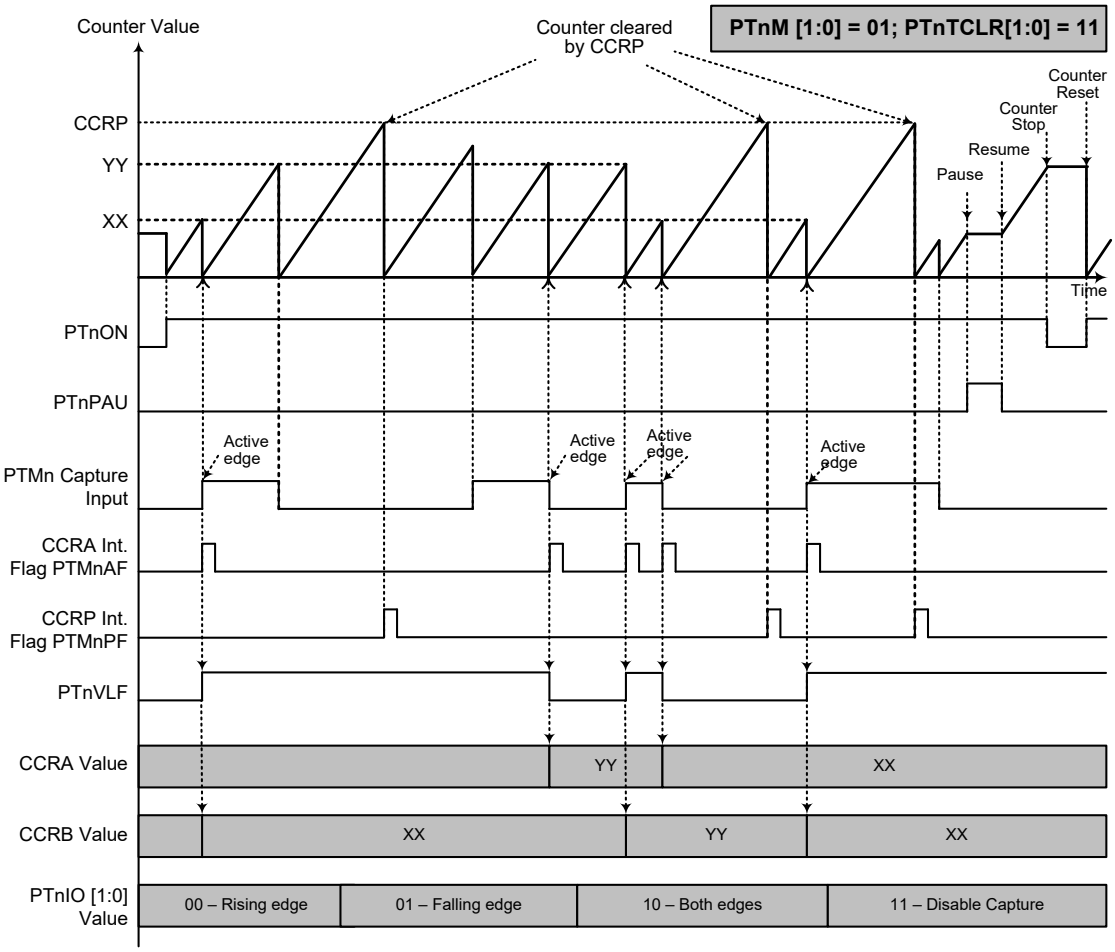
2. A PTMn capture input (PTCKn, PTPnI pin or CXCAP signal) active edge transfers the counter value to CCRA or CCRB
3. Comparator P match or PTMn capture input rising edge will clear the counter
4. PTnCCLR bit is not used
5. No output function – PTnOC and PTnPOL bits are not used
6. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero
7. The capture input mode cannot be used if the selected PTMn counter clock is not available



Capture Input Mode – PTnTCLR[1:0]=10 (n=0)

Note: 1. PTnM[1:0]=01, PTnTCLR[1:0]=10 and active edge set by the PTnIO[1:0] bits

2. A PTMn capture input (PTCKn, PTPnI pin or CXCAP signal) active edge transfers the counter value to CCRA or CCRB
3. Comparator P match or PTMn capture input falling edge will clear the counter
4. PTnCCLR bit is not used
5. No output function – PTnOC and PTnPOL bits are not used
6. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero
7. The capture input mode cannot be used if the selected PTMn counter clock is not available



Capture Input Mode – PTnTCLR[1:0]=11 (n=0)

Note: 1. PTnM[1:0]=01, PTnTCLR[1:0]=11 and active edge set by the PTnIO[1:0] bits

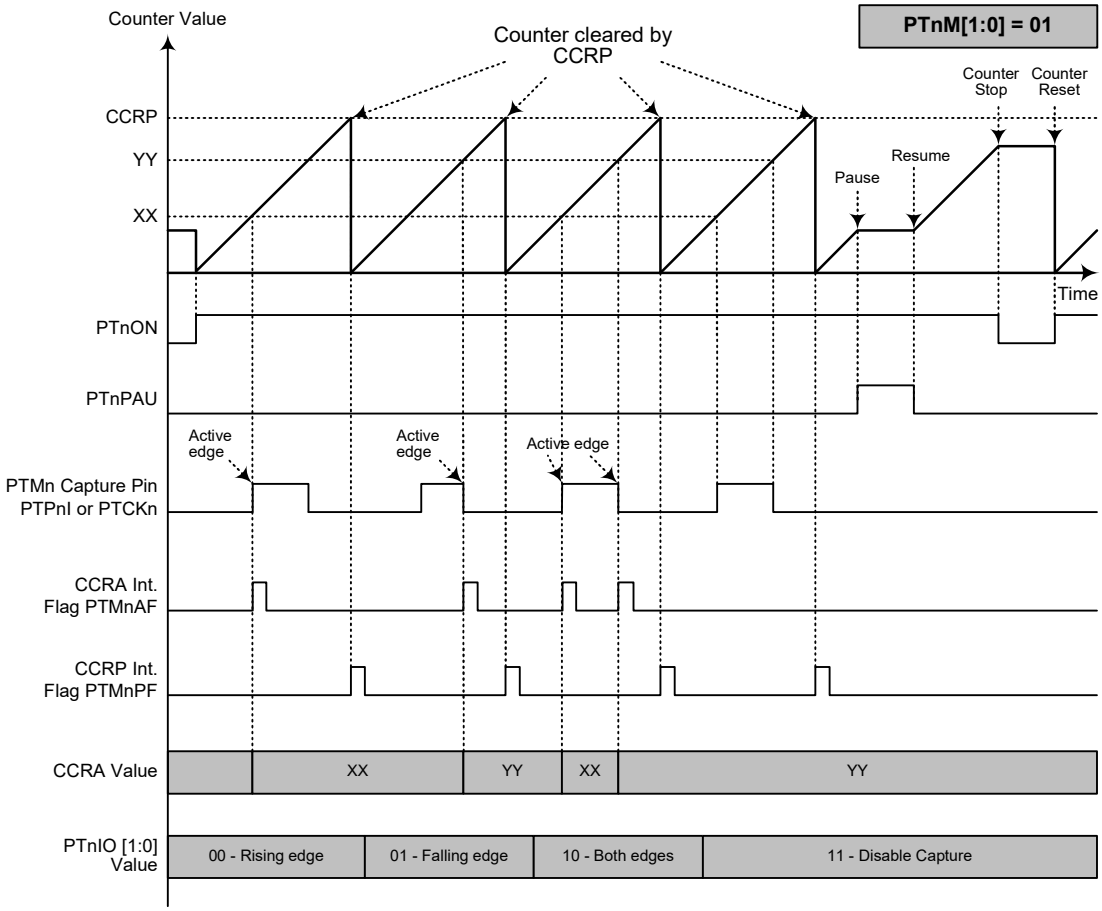
2. A PTMn capture input (PTCKn, PTPnI pin or CXCAP signal) active edge transfers the counter value to CCRA or CCRB
3. Comparator P match or PTMn capture input rising or falling edge will clear the counter
4. PTnCCLR bit is not used
5. No output function – PTnOC and PTnPOL bits are not used
6. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero
7. The capture input mode cannot be used if the selected PTMn counter clock is not available

Capture Input Mode (n=1~2)

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

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

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

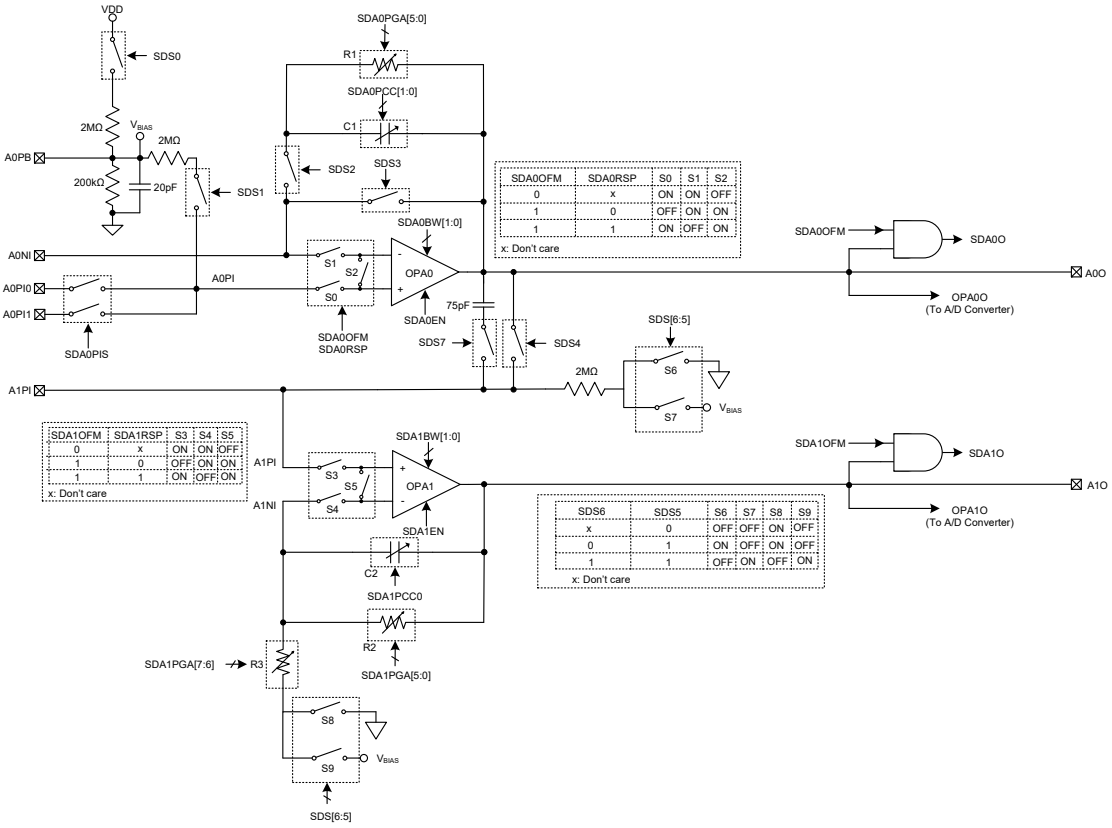


Capture Input Mode (n=1~2)

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

Smoke Detector AFE

The device provides a Smoke Detector AFE circuit which can be used for optical signal detection in Smoke Detector applications. The circuit consists of two fully integrated Operational Amplifiers. The optical signal can be detected and processed by the operational amplifiers.



SDA0OFM	SDA0RSP	S0	S1	S2
0	x	ON	ON	OFF
1	0	OFF	ON	ON
1	1	ON	OFF	ON

SDA1OFM	SDA1RSP	S3	S4	S5
0	x	ON	ON	OFF
1	0	OFF	ON	ON
1	1	ON	OFF	ON

SDS6	SDS5	S6	S7	S8	S9
x	0	OFF	OFF	ON	OFF
0	1	ON	OFF	ON	OFF
1	1	OFF	ON	OFF	ON

Smoke Detector AFE Block Diagram

Note that although the SD OPAn bandwidth is determined by the SDAnBW1~SDAnBW0 bits, there are some limitations when using the OPAn together with the A/D converter. As the OPAn bandwidth will result in a small current output, care must be taken for SD OPAn bandwidths. Refer to the following table for examples, where values marked with a "√" are usable and ensure that the values read by the 12-bit A/D converter are less than 1 LSB.

SD OPAn Bandwidth Selection	A/D Converter Clock Frequency (kHz)							
	15.625	31.25	62.5	125	250	500	1000	2000
SDAnBW[1:0]=00	√	—	—	—	—	—	—	—
SDAnBW[1:0]=01	√	√	√	√	—	—	—	—
SDAnBW[1:0]=10	√	√	√	√	√	√	√	√
SDAnBW[1:0]=11	√	√	√	√	√	√	√	√

Smoke Detector AFE SD OPAn Bandwidth Examples (n=0~1)

Smoke Detector AFE Registers

Overall operation of the Smoke Detector AFE circuit is controlled using a series of registers. The SDSW0 register is used to control the switches on or off thus controlling the OPAn operating mode. The SDSW1 register is used to select the OPA0 positive input channel and the C1, C2 capacitance. The SDPGAC0 and SDPGAC1 registers are used to select the R1, R2 and R3 resistance. The SDAnC register where n=0~1, is used to control the SD OPAn enable/disable and bandwidth functions as well as store the output status. The SDAnVOS register is used to select and control the SD OPAn input offset voltage calibration function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SDSW0	SDS7	SDS6	SDS5	SDS4	SDS3	SDS2	SDS1	SDS0
SDSW1	SDA0PIS	—	SDA0PCC1	SDA0PCC0	—	—	—	SDA1PCC0
SDPGAC0	—	—	SDA0PGA5	SDA0PGA4	SDA0PGA3	SDA0PGA2	SDA0PGA1	SDA0PGA0
SDPGAC1	SDA1PGA7	SDA1PGA6	SDA1PGA5	SDA1PGA4	SDA1PGA3	SDA1PGA2	SDA1PGA1	SDA1PGA0
SDA0C	—	SDA0EN	SDA0O	—	—	—	SDA0BW1	SDA0BW0
SDA1C	—	SDA1EN	SDA1O	—	—	—	SDA1BW1	SDA1BW0
SDA0VOS	SDA0OFM	SDA0RSP	SDA0OF5	SDA0OF4	SDA0OF3	SDA0OF2	SDA0OF1	SDA0OF0
SDA1VOS	SDA1OFM	SDA1RSP	SDA1OF5	SDA1OF4	SDA1OF3	SDA1OF2	SDA1OF1	SDA1OF0

Smoke Detector AFE Register List

• SDSW0 Register

Bit	7	6	5	4	3	2	1	0
Name	SDS7	SDS6	SDS5	SDS4	SDS3	SDS2	SDS1	SDS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	0	0	0	0	0

Bit 7 **SDS7**: SDS7 switch on/off control
0: Off
1: On

Bit 6~5 **SDS6~SDS5**: Mode control
00: External mode
01: AC coupling mode
10: External mode
11: DC coupling mode (SDS1 cannot be on at the same time)

Note: 1.When the Switch S6 and S8 are both on to select the AC couple mode, the OPAMP1 amplification factor will be decreased as the Switch S8 has an on-resistance R_{ON_S8}. The OPAMP1 Gain=1+R2/(R3+R_{ON_S8}), where R_{ON_S8}≈150Ω@V_{DD}=5V, R_{ON_S8}≈170Ω@V_{DD}=3V

2. When the Switch S7 and S9 are both on, connecting to VBIAS, to select the DC couple mode, the original settings of the resistance amplification factor will not be available for reference, which is an especially important point users should pay attention to.

Bit 4 **SDS4**: SDS4 switch on/off control
0: Off
1: On

Bit 3 **SDS3**: SDS3 switch on/off control
0: Off
1: On

Bit 2 **SDS2**: SDS2 switch on/off control
0: Off
1: On

Bit 1 **SDS1**: SDS1 switch on/off control
0: Off
1: On

Bit 0 **SDS0**: SDS0 switch on/off control
0: Off
1: On

• **SDSW1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SDA0PIS	—	SDA0PCC1	SDA0PCC0	—	—	—	SDA1PCC0
R/W	R/W	—	R/W	R/W	—	—	—	R/W
POR	0	—	0	0	—	—	—	0

Bit 7 **SDA0PIS**: SD OPA0 positive input selection
0: A0PI0
1: A0PI1

Bit 6 Unimplemented, read as "0"

Bit 5~4 **SDA0PCC1~SDA0PCC0**: SD OPA0 C1 control
00: 20pF
01: 15pF
10: 10pF
11: 20pF

Bit 3~1 Unimplemented, read as "0"

Bit 0 **SDA1PCC0**: SD OPA1 C2 control
0: 30pF
1: 15pF

• **SDPGAC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SDA0PGA5	SDA0PGA4	SDA0PGA3	SDA0PGA2	SDA0PGA1	SDA0PGA0
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 **SDA0PGA5~SDA0PGA0**: R1 control
R1=(SDA0PGA[5:0]×100kΩ)
These bits are used to select the R1 resistance value. Note that R1≠0Ω when these bits are set to "000000" due to the switch on-resistance.

• **SDPGAC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SDA1PGA7	SDA1PGA6	SDA1PGA5	SDA1PGA4	SDA1PGA3	SDA1PGA2	SDA1PGA1	SDA1PGA0
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 **SDA1PGA7~SDA1PGA6**: R3 control
00: 10kΩ
01: 20kΩ
10: 30kΩ
11: 40kΩ

Bit 5~0 **SDA1PGA5~SDA1PGA0**: R2 control
R2=(SDA1PGA[5:0]×100kΩ)
These bits are used to select the R2 resistance value. Note that R2≠0Ω when these bits are set to "000000" due to the switch on-resistance.

• **SDA0C Register**

Bit	7	6	5	4	3	2	1	0
Name	—	SDA0EN	SDA0O	—	—	—	SDA0BW1	SDA0BW0
R/W	—	R/W	R	—	—	—	R/W	R/W
POR	—	0	0	—	—	—	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **SDA0EN**: SD OPA0 enable or disable control
0: Disable
1: Enable

Bit 5 **SDA0O**: SD OPA0 output status (positive logic)
This bit is read only.
When the SDA0OFM bit is set to 1, SDA0O is defined as SD OPA0 output status, refer to the "Operational Amplifier Input Offset Calibration" section for details.
When the SDA0OFM bit is cleared to 0, this bit will be fixed at a low level.

Bit 4~2 Unimplemented, read as "0"

Bit 1~0 **SDA0BW1~SDA0BW0**: SD OPA0 bandwidth control
00: 5kHz
01: 40kHz
10: 600kHz
11: 2MHz
Refer to "Smoke Detector AFE Electrical Characteristics" for details.

• **SDA1C Register**

Bit	7	6	5	4	3	2	1	0
Name	—	SDA1EN	SDA1O	—	—	—	SDA1BW1	SDA1BW0
R/W	—	R/W	R	—	—	—	R/W	R/W
POR	—	0	0	—	—	—	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **SDA1EN**: SD OPA1 enable or disable control
0: Disable
1: Enable

Bit 5 **SDA1O**: SD OPA1 output status (positive logic)
This bit is read only.
When the SDA1OFM bit is set to 1, SDA1O is defined as SD OPA1 output status, refer to the "Operational Amplifier Input Offset Calibration" section for details.
When the SDA1OFM bit is cleared to 0, this bit will be fixed at a low level.

Bit 4~2 Unimplemented, read as "0"

Bit 1~0 **SDA1BW1~SDA1BW0**: SD OPA1 bandwidth control
00: 5kHz
01: 40kHz
10: 600kHz
11: 2MHz
Refer to "Smoke Detector AFE Electrical Characteristics" for details.

• **SDA0VOS Register**

Bit	7	6	5	4	3	2	1	0
Name	SDA0OFM	SDA0RSP	SDA0OF5	SDA0OF4	SDA0OF3	SDA0OF2	SDA0OF1	SDA0OF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **SDA0OFM**: SD OPA0 normal operation or input offset voltage calibration mode selection
0: Normal operation
1: Offset calibration mode

Bit 6 **SDA0RSP**: SD OPA0 input offset voltage calibration reference selection
0: Input reference voltage comes from A0NI
1: Input reference voltage comes from A0PI

Bit 5~0 **SDA0OF5~SDA0OF0**: SD OPA0 input offset voltage calibration control
This 6-bit field is used to perform the operational amplifier input offset calibration operation and the value for the SD OPA0 input offset Calibration can be restored into this bit field. More detailed information is described in the "Operational Amplifier Input Offset Calibration" section.

• **SDA1VOS Register**

Bit	7	6	5	4	3	2	1	0
Name	SDA1OFM	SDA1RSP	SDA1OF5	SDA1OF4	SDA1OF3	SDA1OF2	SDA1OF1	SDA1OF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **SDA1OFM**: SD OPA1 normal operation or input offset voltage calibration mode selection
0: Normal operation
1: Offset calibration mode

Bit 6 **SDA1RSP**: SD OPA1 input offset voltage calibration reference selection
0: Input reference voltage comes from A1NI
1: Input reference voltage comes from A1PI

Bit 5~0 **SDA1OF5~SDA1OF0**: SD OPA1 input offset voltage calibration control
These 6 bits are used to perform the operational amplifier input offset calibration operation and the value for the SD OPA1 input offset Calibration can be restored into this bit field. More detailed information is described in the "Operational Amplifier Input Offset Calibration" section.

Operational Amplifier Operation

There are two fully integrated Operational Amplifiers in the device, OPA0 and OPA1. These OPAs can be used for signal amplification according to specific user requirements. The OPAs can be disabled or enabled entirely under software control using internal registers. With specific control registers, some OPA related applications can be more flexible and easier to be implemented, such as Unit Gain Buffer, Non-Inverting Amplifier, Inverting Amplifier and various kinds of filters, etc.

Operational Amplifier Input Offset Calibration

Note that if the SD Operational Amplifier inputs are pin-shared with I/O pins, they should be configured as the SD Operational Amplifier input function before the Input Offset Calibration.

Step 1. Set SDAnOFM=1 and SDAnRSP=1, the SD Operational Amplifier n is now under the input offset Calibration mode, S0 and S2 on. To make sure the V_{AnOS} as minimal as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal operation.

Step 2. Set SDAnOF[5:0]=000000 and then read the SDAnO bit.

Step 3. Increase the SDAnOF[5:0] value by 1 and then read the SDAnO bit.

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

If the SDAnO bit state has changed, record the SDAnOF[5:0] value as V_{AnOS1} and then go to Step 4.

Step 4. Set SDAnOF[5:0]=111111 and read the SDAnO bit.

Step 5. Decrease the SDAnOF[5:0] value by 1 and then read the SDAnO bit.

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

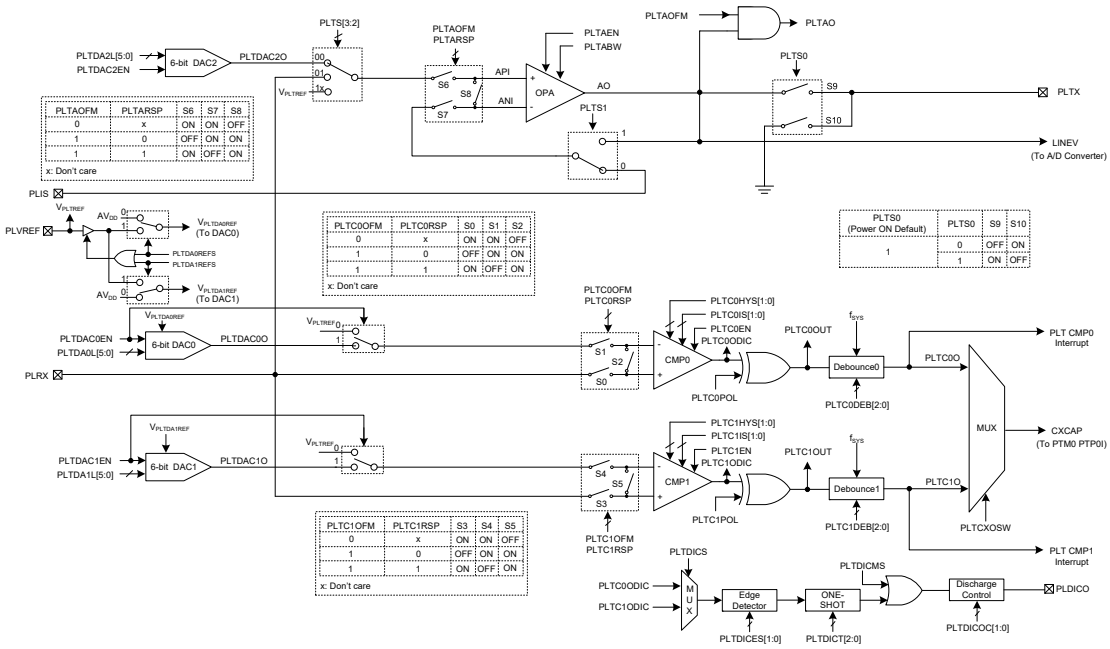
If the SDAnO bit state has changed, record the SDAnOF[5:0] value as V_{AnOS2} and then go to Step 6.

Step 6. Restore the SD Operational Amplifier n input offset calibration value V_{AnOS} into the SDAnOF[5:0] bits. The offset Calibration procedure is now finished.

$V_{AnOS}=(V_{AnOS1}+V_{AnOS2})/2$. If $(V_{AnOS1}+V_{AnOS2})/2$ is not integral, discard the decimal.

Power Line Transceiver – PLT

The device provides a power line transceiver circuit which can be used for power line data transmission and reception. The circuit consists of three 6-bit D/A Converters, one fully integrated Operational Amplifier and two Comparators. The reference voltage of the DAC0/DAC1 is selected by the PLTDA0REFS/PLTDA1REFS bit. If the bit is cleared to 0, the AV_{DD} is seleted; if the bit is set to 1, the V_{PLTREF} is selected. Note that the PLVREF pin is pin-shared with other functions, it should be properly configure the corresponding pin-shared function control bits before using the V_{PLTREF} voltage.



Note: 1. After the ONE-SHOT circuit is triggered, it will not be triggered repeatedly during its operation.
2. When the PLTDICMS bit is equal to 0 and the ONE-SHOT circuit is not triggered, the PLDICO pin is high-impadance.
3. When the ONE-SHOT circuit is being triggered, PLTDICT[2:0] must not be modified to avoid uncertain output state.

Power Line Transceiver Block Diagram

Power Line Transceiver Registers

Overall operation of the Power Line Transceiver circuit is controlled using a series of registers. The DACn outputs, the Operational Amplifier, Comparator input signal selection, operating modes, output signals all can be setup using these registers by application program.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PLTSW	—	—	—	—	PLTS3	PLTS2	PLTS1	PLTS0
PLTDACC	—	—	—	PLTDA1REFS	PLTDA0REFS	PLTDAC2EN	PLTDAC1EN	PLTDAC0EN
PLTDA0L	—	—	D5	D4	D3	D2	D1	D0
PLTDA1L	—	—	D5	D4	D3	D2	D1	D0
PLTDA2L	—	—	D5	D4	D3	D2	D1	D0
PLTC0C	PLTC0OUT	PLTC0EN	PLTC0O	PLTC0DEB2	PLTC0DEB1	PLTC0DEB0	PLTC0IS1	PLTC0IS0
PLTC1C	PLTC1OUT	PLTC1EN	PLTC1O	PLTC1DEB2	PLTC1DEB1	PLTC1DEB0	PLTC1IS1	PLTC1IS0
PLTC0VOS	—	PLTC0OFM	PLTC0RSP	PLTC0OF4	PLTC0OF3	PLTC0OF2	PLTC0OF1	PLTC0OF0
PLTC1VOS	—	PLTC1OFM	PLTC1RSP	PLTC1OF4	PLTC1OF3	PLTC1OF2	PLTC1OF1	PLTC1OF0
PLTCHYC	—	PLTCXOSW	PLTC1POL	PLTC0POL	PLTC1HYS1	PLTC1HYS0	PLTC0HYS1	PLTC0HYS0
PLTAC	—	PLTAEN	PLTAO	—	—	—	—	PLTABW
PLTAVOS	PLTAOFM	PLTARSP	PLTAOF5	PLTAOF4	PLTAOF3	PLTAOF2	PLTAOF1	PLTAOF0
PLTDICC0	PLTDICMS	—	—	—	—	—	PLTDICOC1	PLTDICOC0
PLTDICC1	PLTDICS	PLTDICES1	PLTDICES0	—	—	PLTDICT2	PLTDICT1	PLTDICT0

Power Line Transceiver Register List

• PLTSW Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PLTS3	PLTS2	PLTS1	PLTS0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **PLTS3~PLTS2**: PLTS[3:2] switch selection
- 00: Connect to PLTDAC2O
- 01: Connect to PLRX
- 1x: Connect to PLVREF

Bit 1 **PLTS1**: PLTS1 switch selection
- 0: Connect to PLIS
- 1: Connect to LINEV

Bit 0 **PLTS0**: PLTX switch selection
- 0: PLTX switch to GND
- 1: PLTX switch to AO

• PLTDACC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	PLTDA1REFS	PLTDA0REFS	PLTDAC2EN	PLTDAC1EN	PLTDAC0EN
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 **PLTDA1REFS**: PLT DAC1 reference voltage ($V_{PLTDA1REF}$) selection
- 0: AV_{DD}
- 1: V_{PLTREF}

Bit 3 **PLTDA0REFS**: PLT DAC0 reference voltage ($V_{PLTDA0REF}$) selection
- 0: AV_{DD}
- 1: V_{PLTREF}

Bit 2 **PLTDAC2EN**: PLT DAC2 enable or disable control
- 0: Disable (PLTDAC2O high impedance)
- 1: Enable

Bit 1 **PLTDAC1EN**: PLT DAC1 enable or disable control
0: Disable (CMP1 negative input is V_{PLTREF})
1: Enable (CMP1 negative input is PLTDAC1O)

Bit 0 **PLTDAC0EN**: PLT DAC0 enable or disable control
0: Disable(CMP0 negative input is V_{PLTREF})
1: Enable (CMP0 negative input is PLTDAC0O)

• PLTDA0L 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**: PLT DAC0 output control code
PLTDAC0O=(DAC $V_{PLTDA0REF}/2^6$)×PLTDA0L[5:0]

• PLTDA1L 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**: PLT DAC1 output control code
PLTDAC1O=(DAC $V_{PLTDA1REF}/2^6$)×PLTDA1L[5:0]

• PLTDA2L 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**: PLT DAC2 output control code
PLTDAC2O=(DAC $AV_{DD}/2^6$)×PLTDA2L[5:0]

• PLTC0C Register

Bit	7	6	5	4	3	2	1	0
Name	PLTC0OUT	PLTC0EN	PLTC0O	PLTC0DEB2	PLTC0DEB1	PLTC0DEB0	PLTC0IS1	PLTC0IS0
R/W	R	R/W	R	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PLTC0OUT**: PLT Comparator 0 output bit
If PLTC0POL=0 and input voltages of the comparator are
C0PI > C0NI → PLTC0OUT=1
C0NI > C0PI → PLTC0OUT=0
If PLTC0POL=1 and input voltages of the comparator are
C0PI < C0NI → PLTC0OUT=1
C0NI < C0PI → PLTC0OUT=0

Bit 6 **PLTC0EN**: PLT Comparator 0 enable or disable control
0: Comparator disable
1: Comparator enable
This is the PLT Comparator 0 on/off control bit. If the comparator is disabled, the comparator output will be cleared to 0. Therefore, PLTC0OUT=0 when PLTC0POL=0, or PLTC0OUT=1 when PLTC0POL=1.

Bit 5 **PLTC0O**: PLT Comparator 0 debounced output

The PLTC0O is the de-bounce version of PLTC0OUT.

The PLT CMP0 interrupt will occur when the PLTC0O changes from low to high.

If PLTC0POL=0, the PLTC0O outputs "1" only when the current and previous N samples of PLTC0OUT are "1".

If PLTC0POL=1, The PLTC0O outputs "0" only when the current and previous N samples of PLTC0OUT are "0". N depends on PLTC0DEB [2:0] configuration bits.

Bit 4~2 **PLTC0DEB2~PLTC0DEB0**: PLT Comparator 0 debounce time control

000: No debounce
001: $2^1 \times t_{SYS}$
010: $2^2 \times t_{SYS}$
011: $2^3 \times t_{SYS}$
100: $2^4 \times t_{SYS}$
101: $2^5 \times t_{SYS}$
110: $2^6 \times t_{SYS}$
111: $2^7 \times t_{SYS}$

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

Bit 1~0 **PLTC0IS1~PLTC0IS0**: PLT Comparator 0 current control

Refer to the "Comparator Electrical Characteristics" table for details.

- **PLTC1C Register**

Bit	7	6	5	4	3	2	1	0
Name	PLTC1OUT	PLTC1EN	PLTC1O	PLTC1DEB2	PLTC1DEB1	PLTC1DEB0	PLTC1IS1	PLTC1IS0
R/W	R	R/W	R	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 PLTC1OUT: PLT Comparator 1 output bit

If PLTC1POL=0 and input voltages of the comparator are
C1PI>C1NI → PLTC1OUT=1
C1NI>C1PI → PLTC1OUT=0

If PLTC1POL=1 and input voltages of the comparator are
C1PI<C1NI → PLTC1OUT=1
C1NI<C1PI → PLTC1OUT=0

Bit 6 **PLTC1EN**: PLT Comparator 1 enable or disable control

0: Comparator disable
1: Comparator enable

This is the PLT Comparator 1 on/off control bit. If the comparator is disabled, the comparator output will be cleared to 0. Therefore, PLTC1OUT=0 when PLTC1POL=0, or PLTC1OUT=1 when PLTC1POL=1.

Bit 5 **PLTC1O**: PLT Comparator 1 debounced output

The PLTC1O is the de-bounce version of PLTC1OUT.

The PLT CMP1 interrupt will occur when the PLTC1O changes from low to high.

If PLTC1POL=0, the PLTC1O outputs "1" only when the current and previous N samples of PLTC1OUT are "1".

If PLTC1POL=1, The PLTC1O outputs "0" only when the current and previous N samples of PLTC1OUT are "0". N depends on PLTC1DEB [2:0] configuration bits.

Bit 4~2 **PLTC1DEB2~PLTC1DEB0**: PLT Comparator 1 debounce time control

000: No debounce
001: $2^1 \times t_{SYS}$
010: $2^2 \times t_{SYS}$
011: $2^3 \times t_{SYS}$
100: $2^4 \times t_{SYS}$
101: $2^5 \times t_{SYS}$
110: $2^6 \times t_{SYS}$
111: $2^7 \times t_{SYS}$

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

Bit 1~0 **PLTC1IS1~PLTC1IS0**: PLT Comparator 1 current control
Refer to the "Comparator Electrical Characteristics" table for details.

• **PLTC0VOS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PLTC0OFM	PLTC0RSP	PLTC0OF4	PLTC0OF3	PLTC0OF2	PLTC0OF1	PLTC0OF0
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 Unimplemented, read as "0"

Bit 6 **PLTC0OFM**: PLT Comparator 0 normal operation or input offset voltage calibration mode selection
0: Normal operation
1: Offset calibration mode

Bit 5 **PLTC0RSP**: PLT Comparator 0 input offset voltage calibration reference selection
0: Input reference voltage comes from C0NI
1: Input reference voltage comes from C0PI

Bit 4~0 **PLTC0OF4~PLTC0OF0**: PLT Comparator 0 input offset voltage calibration control
This 5-bit field is used to perform the PLT comparator 0 input offset calibration operation and the value for the PLT Comparator 0 input offset calibration can be restored into this bit field. More detailed information is described in the "Comparator Input Offset Calibration" section.

• **PLTC1VOS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PLTC1OFM	PLTC1RSP	PLTC1OF4	PLTC1OF3	PLTC1OF2	PLTC1OF1	PLTC1OF0
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 Unimplemented, read as "0"

Bit 6 **PLTC1OFM**: PLT Comparator 1 normal operation or input offset voltage calibration mode selection
0: Normal operation
1: Offset calibration mode

Bit 5 **PLTC1RSP**: PLT Comparator 1 input offset voltage calibration reference selection
0: Input reference voltage comes from C1NI
1: Input reference voltage comes from C1PI

Bit 4~0 **PLTC1OF4~PLTC1OF0**: PLT Comparator 1 input offset voltage calibration control
This 5-bit field is used to perform the PLT comparator 1 input offset calibration operation and the value for the PLT Comparator 1 input offset calibration can be restored into this bit field. More detailed information is described in the "Comparator Input Offset Calibration" section.

• **PLTCHYC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PLTCXOSW	PLTC1POL	PLTC0POL	PLTC1HYS1	PLTC1HYS0	PLTC0HYS1	PLTC0HYS0
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 **PLTCXOSW**: Comparator 0 or Comparator 1 output selection
0: Comparator 0 Output
1: Comparator 1 Output
This is the Comparator 0 or Comparator 1 output selection bit. If the bit is zero, then the PLTC0O bit will be output indicating the output condition of the comparator 0.

If the bit is high the comparator PLTC1O bit will be output indicating the output condition of the comparator 1.

Bit 5 **PLTC1POL**: PLT Comparator 1 output polarity control
0: Non-invert
1: Invert
This is the PLT Comparator 1 polarity bit. If the bit is zero, then the PLTC1OUT bit will reflect the non-inverted output condition of the comparator. If the bit is high the comparator PLTC1OUT bit will be inverted.

Bit 4 **PLTC0POL**: PLT Comparator 0 output polarity control
0: Non-invert
1: Invert
This is the PLT Comparator 0 polarity bit. If the bit is zero, then the PLTC0OUT bit will reflect the non-inverted output condition of the comparator 0. If the bit is high the comparator PLTC0OUT bit will be inverted.

Bit 3~2 **PLTC1HYS1~PLTC1HYS0**: PLT Comparator 1 hysteresis voltage window control
Refer to "Comparator Characteristics" table for details.

Bit 1~0 **PLTC0HYS1~PLTC0HYS0**: PLT Comparator 0 hysteresis voltage window control
Refer to "Comparator Characteristics" table for details.

• **PLTAC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PLTAEN	PLTAO	—	—	—	—	PLTABW
R/W	—	R/W	R	—	—	—	—	R/W
POR	—	0	0	—	—	—	—	0

Bit 7 Unimplemented, read as "0"

Bit 6 **PLTAEN**: PLT OPA enable or disable control
0: Disable (AO high impedenece)
1: Enable

Bit 5 **PLTAO**: PLT OPA output status (positive logic)
This bit is read only.
When the PLTAOFM bit is set to 1, PLTAO is defined as PLT OPA output status, refer to "Operational Amplifier Input Offset Calibration". When the PLTAOFM bit is cleared to 0, this bit will be fixed at a low level.

Bit 4~1 Unimplemented, read as "0"

Bit 0 **PLTABW**: PLT OPA Gain bandwith control bit
0: 600kHz
1: 2MHz
Refer to "Operational Amplifier Electrical Characteristics" table for details.

• **PLTAVOS Register**

Bit	7	6	5	4	3	2	1	0
Name	PLTAOFM	PLTARSP	PLTAOF5	PLTAOF4	PLTAOF3	PLTAOF2	PLTAOF1	PLTAOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **PLTAOFM**: PLT OPA normal operation or input offset voltage cancellation mode selection
0: Normal operation
1: Offset calibration mode

Bit 6 **PLTARSP**: PLT OPA input offset voltage calibration reference selection
0: Input reference voltage comes from ANI
1: Input reference voltage comes from API

Bit 5~0 **PLTAOF5~PLTAOF0**: PLT OPA input offset voltage calibration control

This 6-bit field is used to perform the PLT OPA input offset calibration operation and the value for the PLT OPA input offset calibration can be restored into this bit filed. More detailed information is described in the "Operational Amplifier Input Offset Calibration" section.

• **PLTDICC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PLTDICMS	—	—	—	—	—	PLTDICOC1	PLTDICOC0
R/W	R/W	—	—	—	—	—	R/W	R/W
POR	0	—	—	—	—	—	0	0

Bit 7 **PLTDICMS**: PLT discharge mode selection
0: Automatic discharge mode
1: Manual discharge mode

When this bit is set high, the discharge capacity will be selected by PLTDICOC1~PLTDICOC0 bits.

Bit 6~2 Unimplemented, read as "0"

Bit 1~0 **PLTDICOC1~PLTDICOC0**: PLT discharge capacity selection
00: 75μA
01: 150μA
10: 300μA
11: 600μA

• **PLTDICC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PLTDICS	PLTDICES1	PLTDICES0	—	—	PLTDICT2	PLTDICT1	PLTDICT0
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7 **PLTDICS**: PLT discharge function trigger source selection
0: PLTC0ODIC
1: PLTC1ODIC

Bit 6~5 **PLTDICES1~PLTDICES0**: PLT active edge selection
00: Disable
01: Falling edge
10: Rising edge
11: Both falling and rising edges

Bit 4~3 Unimplemented, read as "0"

Bit 2~0 **PLTDICT2~PLTDICT0**: PLT discharge time selection
000: $2^1 \times t_{SYS}$
001: $2^2 \times t_{SYS}$
010: $2^3 \times t_{SYS}$
011: $2^4 \times t_{SYS}$
100: $2^5 \times t_{SYS}$
101: $2^6 \times t_{SYS}$
110: $2^7 \times t_{SYS}$
111: $2^1 \times t_{SYS}$

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

Discharge Circuit Operation

There is a discharge circuit in this device. The discharge trigger source is selected by the PLTDICS bit. The active edge of trigger signal can be disabled, a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PLTDICES1 and PLTDICES0 bits. The ONE-SHOT circuit is initiated by the trigger signal. The discharge capacity is set by the PLTDICOC1 and PLTDICOC0 bits; the discharge time is set by the PLTDICT[2:0] bits.

In addition, the discharge function can also be initiated through software, set PLTDICMS to 1, and set the discharge capacity according to PLTDICOC1 and PLTDICOC0 bits until PLTDICMS is cleared to 0 to disable the discharge function.

Offset Calibration Procedure

To operate in the input offset calibration mode for the PLT Operational Amplifier or the Comparators, the PLTAOFM or PLTCnOFM bit should first be set to "1" to select the input offset voltage calibration mode. Note that as the comparator or Operational Amplifier input is from the PLRX or PLIS pin which is pin-shared with I/O or other functions, before the calibration, they should be configured as PLT comparator or operational amplifier input pin function first.

Comparator Input Offset Calibration

Step 1. Set PLTCnOFM=1 and PLTCnRSP=1, the PLT Comparator n is now operating in the comparator input offset calibration mode, S0 and S2 on or S3 and S5 on. To make sure V_{CnOS} as minimal as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal operation.

Step 2. Set PLTCnOF[4:0]=00000 and read the PLTCnOUT bit.

Step 3. Increase the PLTCnOF[4:0] value by 1 and then read the PLTCnOUT bit.

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

If the PLTCnOUT bit state has changed, record the PLTCnOF[4:0] value as V_{CnOS1} and then go to Step 4.

Step 4. Set PLTCnOF[4:0]=11111 and then read the PLTCnOUT bit.

Step 5. Decrease the PLTCnOF[4:0] value by 1 and then read the PLTCnOUT bit.

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

If the PLTCnOUT bit state has changed, record the PLTCnOF[4:0] value as V_{CnOS2} and then go to Step 6.

Step 6. Restore the PLT Comparator n input offset calibration value V_{CnOS} into the PLTCnOF[4:0] bit field. The offset Calibration procedure is now finished.

$V_{CnOS}=(V_{CnOS1}+V_{CnOS2})/2$. If $(V_{CnOS1}+V_{CnOS2})/2$ is not integral, discard the decimal.

Operational Amplifier Input Offset Calibration

Step 1. Set PLTAOFM=1 and PTLARSP=1, the PLT Operational Amplifier is now under offset calibration mode, S6 and S8 on. To make sure V_{AOS} as minimal as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal mode operation.

Step 2. Set PTLAOF[5:0]=000000 and then read PTLAO bit

Step 3. Increase the PLTAOF[5:0] value by 1 and then read the PLTAO bit.

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

If the PLTAO bit state has changed, record the PLTAOF[5:0] value as V_{AOS1} and then go to Step 4.

Step 4. Set PTLAOF[5:0]=111111 then read PLTAO bit.

Step 5. Decrease the PLTAOF[5:0] value by 1 and then read the PLTAO bit.

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

If the PLTAO bit state has changed, record the PLTAOF[5:0] value as V_{AOS2} and then go to Step 6.

Step 6. Restore the PLT Operational Amplifier input offset calibration value V_{AOS} into the PLTAOF[5:0] bit field. The offset Calibration procedure is now finished.

$V_{AOS}=(V_{AOS1}+V_{AOS2})/2$. If $(V_{AOS1}+V_{AOS2})/2$ is not integral, discard the decimal.

Analog to Digital Converter

The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the microcontroller, the need for external components is reduced significantly with the corresponding follow-on benefits of lower costs and reduced component space requirements.

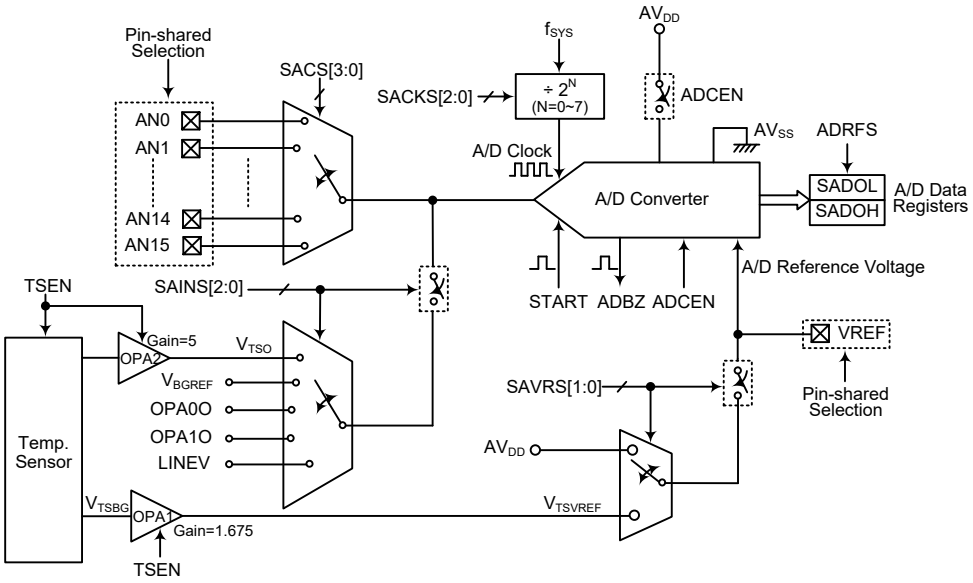
A/D Converter Overview

The device contains a multi-channel 12-bit analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, the high performance bandgap reference voltage V_{BGREF}, the SD operational amplifier 0 output signal OPA0O and the SD operational amplifier 1 output signal OPA1O, PLT operational amplifier output signal, LINEV, or the temperature sensor output voltage, V_{TSO}, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 bits together with the SACS3~SACS0 bits. Note that when the internal analog signal is to be converted, the SAINS2~SAINS0 and relevant pin-shared control bits should be properly configured to avoid external channel input. More detailed information about the A/D input signal is described in the “A/D Converter Control Registers” and “A/D Converter Input Signals” sections respectively.

This A/D converter also includes a temperature sensor circuitry which contains a temperature sensor, two operational amplifiers and an internal reference voltage. The temperature sensor can detect the temperature and then output a voltage proportional to the temperature. The output voltage can be amplified by the OPA and then converted to a 12-bit digital data using the A/D converter.

External Input Channels	Internal Signals	Channel Select Bits
16: AN0~AN15	5: V_{BGREF}, OPA0O, OPA1O, LINEV, V_{TSO}	SAINS2~SAINS0, SACS3~SACS0

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



A/D Converter with temperature sensor Structure

A/D Converter Register Description

Overall operation of the A/D converter with temperature sensor is controlled using several registers. A read only register pair exists to store the A/D converter data 12-bit value. Two registers, SADC0 and SADC1, are control registers which setup the operating and control function of the A/D converter. The SADC2 register is used to enable/disable the integrated temperature sensor circuitry and select the A/D converter conversion mode. The VBGRC resigter is used to enable/disable the A/D converter internal bandgap reference voltage ouput.

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

A/D Converter with temperature sensor Register List

A/D Converter Data Registers – SADOL, SADOH

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

unused bits will be read as zero. Note that A/D Converter data register contents will be unchanged if the A/D converter is disabled.

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

A/D Converter Data Registers

A/D Converter Control Registers – SADC0, SADC1

To control the function and operation of the A/D converter, two control registers known as SADC0 and SADC1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external or internal analog signal inputs must be routed to the converter. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input.

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

- **SADC0 Register**

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

Bit 7 **START**: Start the A/D conversion
0→1→0: Start
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process.

Bit 6 **ADBZ**: A/D converter busy flag
0: No A/D conversion is in progress
1: A/D conversion is in progress
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.

Bit 5 **ADCEN**: A/D converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set high to enable the A/D converter. If the bit is set low, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOH and SADOL will be unchanged.

Bit 4 **ADRFS**: A/D converter data format selection
0: A/D converter data format → SADOH=D[11:4]; SADOL=D[3:0]
1: A/D converter data format → SADOH=D[11:8]; SADOL=D[7:0]
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.

Bit 3~0 **SACS3~SACS0**: A/D converter external analog channel input selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000: AN8
1001: AN9
1010: AN10
1011: AN11
1100: AN12
1101: AN13
1110: AN14
1111: AN15

• SADC1 Register

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
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 7~5 **SAINS2~SAINS0**: A/D converter input signal selection
000: External input – External analog channel input
001: Internal input – Internal high performance bandgap reference voltage, V_{BGREF}
010: Internal input – Internal temperature Sensor output voltage, V_{TSO}
011: Internal input – Internal SD operational amplifier 0 output signal, OPA0O
100: Internal input – Internal SD operational amplifier 1 output signal, OPA1O
101: Internal input – Internal PLT operational amplifier output signal, LINEV
110: External input – External analog channel input
111: Forbidden data, SAINS2~SAINS0 bits cannot be written with "111"
Care must be taken if the SAINS2~SAINS0 bits are set from "001~101" to select the internal analog signal to be converted. When the internal analog signal is selected to be converted, the external input pin must never be selected as the A/D input signal by properly setting the relevant pin-shared control bits. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage selection
00: From external VREF pin
01: Internal A/D converter power, AV_{DD}
10: Internal temperature sensor reference voltage, V_{TSVREF}
11: Internal A/D converter power, AV_{DD}
These bits are used to select the A/D converter reference voltage. Care must be taken if the SAVRS1~SAVRS0 bits are set to "01~11" to select the internal A/D converter power or temperature sensor reference voltage as the reference voltage source. When the internal A/D converter power or temperature sensor reference voltage is selected as the reference voltage, the VREF pin cannot be configured as the reference voltage input by properly configuring the corresponding pin-shared function control bits. Otherwise, the external input voltage on VREF pin will be connected to the internal A/D converter power.

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source select
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$
These three bits are used to select the clock source for the A/D converter.

Bandgap Referenc Voltage Control Register – VBGRC

A high performance bandgap voltage reference is included in the device. It has an accurate voltage reference output, V_{BGREF}, when input supply voltage change or temperature variation. The VBGRC register is used to control the bandgap reference voltage circuit enable or disable.

- **VBGRC Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **VBGREN**: Bandgap enable/disable control
0: Disable
1: Enable
This bit is used to enable/disable the internal Bandgap reference circuit. The internal Bandgap reference circuit should first be enabled before the V_{BGREF} voltage is selected to be used. A specific start-up time is necessary for the Bandgap circuit to become stable and accurate.
When the VBGREN bit is cleared to zero, the Bandgap voltage output is in a high impedance state.

Temperature Sensor Control Register – SADC2

The SADC2 register is used to enable/disable the integrated temperature sensor circuitry and select the A/D converter conversion mode.

- **SADC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAMS	—	—	—	—	—	D1	TSEN
R/W	R/W	—	—	—	—	—	R/W	R/W
POR	0	—	—	—	—	—	1	0

Bit 7 **SAMS**: A/D converter conversion mode selection
0: Normal mode, A/D converter clock rate up to 2MHz
1: Low current mode
This bit is used to select the A/D converter conversion mode. The low current mode A/D converter is suitable for continuous static signal conversion, and its clock frequency range is limited to 100kHz~250kHz.

Bit 6~2 Unimplemented, read as "0"

Bit 1 **D1**: Reserved bit, should be fixed to 1

Bit 0 **TSEN**: Temperature sensor circuitry enable control
0: Disable
1: Enable

This bit controls the internal temperature sensor circuitry. If the temperature sensor output will be converted or the temperature sensor reference voltage will be selected as the A/D conversion reference voltage, the temperature sensor circuitry should be turned on by setting the TSEN bit high first. When the temperature sensor is enabled by setting the TSEN bit to 1, a time named as t_{TSS} should be allowed for the temperature sensor circuit to stabilise before implementing relevant temperature sensor operation.

A/D Converter Operation

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

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

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the SACKS2~SACKS0 bits in the SADC1 register. Although the A/D clock source is determined by the system clock f_{SYS} and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken for system clock frequencies. For example, as the system clock operates at a frequency of 8MHz, the SACKS2~SACKS0 bits should not be set to 000, 001 or 111. Doing so will give A/D clock periods that are less or larger than the minimum or maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where special care must be taken, as the values may be less or larger than the specified A/D Clock Period range.

If the input signal to be converted is the temperature sensor output voltage, the permissible A/D clock period is from 1μs to 2μs.

f_{SYS}	A/D Clock Period (t_{ADCK})							
	SACKS[2:0] = 000 (f_{SYS})	SACKS[2:0] = 001 ($f_{SYS}/2$)	SACKS[2:0] = 010 ($f_{SYS}/4$)	SACKS[2:0] = 011 ($f_{SYS}/8$)	SACKS[2:0] = 100 ($f_{SYS}/16$)	SACKS[2:0] = 101 ($f_{SYS}/32$)	SACKS[2:0] = 110 ($f_{SYS}/64$)	SACKS[2:0] = 111 ($f_{SYS}/128$)
1MHz	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *	128μs *
2MHz	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *
4MHz	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *
8MHz	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *
12MHz	83ns *	167ns *	333ns *	667ns	1.33μs	2.67μs	5.33μs	10.67μs *
16MHz	62.5ns *	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs

A/D Clock Period Examples

Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs, if the ADCEN bit is high, then some power will still be

consumed. In power conscious applications it is therefore recommended that the ADCEN is set low to reduce power consumption when the A/D converter function is not being used.

The SAMS bit in the SADC2 register is used to select the A/D converter conversion mode. The low current mode A/D converter is suitable for continuous static signal conversion, and its clock rate range is limited to 100kHz~250kHz.

A/D Converter Reference Voltage

The reference voltage supply to the A/D converter can be supplied from the power supply AV_{DD}, or from the temperature sensor reference voltage, V_{TSVREF}, or from an external reference source supplied on pin VREF. The desired selection is made using the SAVRS1~SAVRS0 bits. When the SAVRS1~SAVRS0 bits are set to "01" or "11", the A/D converter reference voltage will come from the AV_{DD}. If the temperature sensor reference voltage is required to use, the SAVRS1~SAVRS0 bits should be set to "10". As the temperature sensor circuitry is controlled by the TSEN bit in the SADC2 register, the TSEN bit should be set high to enable the temperature sensor. Otherwise, if the SAVRS1~SAVRS0 bits are set to "00", the A/D converter reference voltage will come from the VREF pin. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bit should be properly configured to disable other pin function. However, if the internal A/D converter power AV_{DD} or temperature sensor reference voltage is selected as the reference voltage, the VREF pin must not be configured as the reference voltage input function to avoid the internal connection between the VREF pin and the power supply. The analog input values must not be allowed to exceed the value of the selected A/D reference voltage.

SAVRS[1:0]	Reference	Description
00	VREF pin	External A/D converter reference pin VREF
01, 11	AV_{DD}	Internal A/D converter power supply voltage
10	V_{TSVREF}	Internal Temperature Sensor reference voltage, V_{TSVREF}

A/D Converter Reference Voltage Selection

A/D Converter Input Signals

All the external A/D analog channel input pins are pin-shared with the I/O pins as well as other functions. The corresponding control bits for each A/D external input pin in the PxS0 and PxS1 registers determine whether the input pins are setup as A/D converter analog input channel or whether they have other functions. If the pin is setup to be as an A/D analog channel input, the original pin functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

There are some internal analog signals derived from the high performance bandgap reference voltage, V_{BGREF}, the SD operational amplifier 0 output signal OPA0O and the SD operational amplifier 1 output signal OPA1O and PLT operational amplifier output signal, LINEV, or the temperature sensor output voltage which can be connected to the A/D converter as the analog input signal by configuring the SAINS2~SAINS0 bits. If the external channel input is selected to be converted, the SAINS2~SAINS0 bits should be set to "000" or "110" and the SACS3~SACS0 bits can determine which external channel is selected. If the internal analog signal is selected to be converted, the relevant pin-shared function selection bits must not be set to select an ANx (x=0~15) pin. Otherwise, the internal analog signal will be connected together with the external channel input. This will result in unpredictable situations.

This V_{BGREF} is the internal high performance bandgap voltage reference with driver capability. It has accurate voltage reference output when input supply voltage AV_{DD} change or temperature variation, and this bandgap will startup at a low supply voltage. Therefore, this voltage reference has high power supply rejection ratio (PSRR) for low dropout regulator (LDO) is presented.

SAINS[2:0]	SACS[3:0]	Input Signals	Description
000, 110	0000~1111	AN0~AN15	External pin analog input
001	xxxx	V_{BGREF}	Internal high performance Bandgap reference voltage
010	xxxx	V_{TSO}	Internal temperature sensor output voltage
011	xxxx	OPA0O	Internal SD operational amplifier 0 output signal
100	xxxx	OPA1O	Internal SD operational amplifier 1 output signal
101	xxxx	LINEV	Internal PLT operational amplifier output signal
111	Forbidden data, SAINS2~SAINS0 bits cannot be written with “111”		

“x”: Don’t care

A/D Converter Input Signal Selection

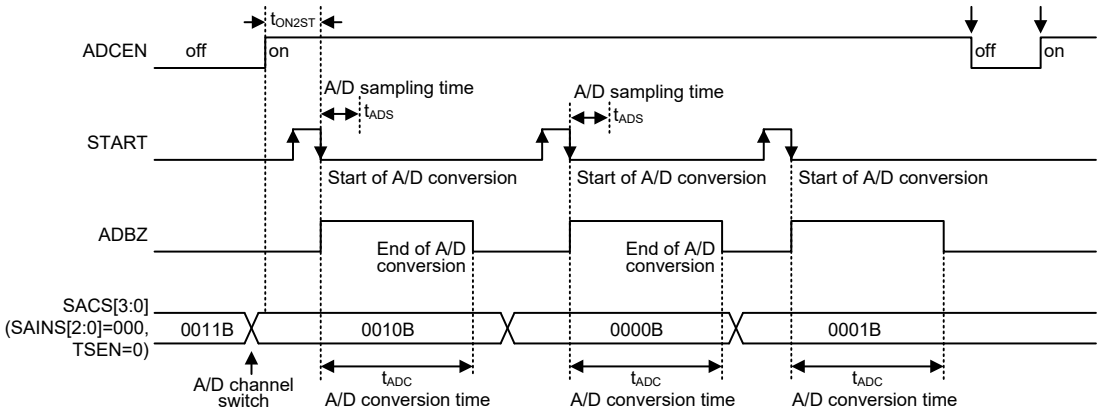
Conversion Rate and Timing Diagram

A complete A/D conversion contains two parts, data sampling and data conversion. If the conversion input signal is not the temperature sensor output, the data sampling which is defined as t_{ADS} takes 4 A/D clock periods and the data conversion takes 12 A/D clock periods. Therefore a total of 16 A/D clock periods for an A/D conversion which is defined as t_{ADC} are necessary. However, an A/D conversion for an internal temperature sensor signal will take a total of 58 A/D clock periods, which includes 46 A/D clock periods for data sampling and 12 A/D clock periods for data conversion.

Maximum single A/D conversion rate=1/(A/D clock period×16) (Temperature sensor output signal is not used)

Maximum single A/D conversion rate=1/(A/D clock period×58) (Temperature sensor output signal is used)

The accompanying diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions.



A/D Conversion Timing – External Channel Input

Summary of A/D Conversion Steps

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

- Step 1

  Select the required A/D conversion clock and A/D conversion mode by correctly programming bits SACKS2~SACKS0 in the SADC1 register and SAMS bit in the SADC2 register.

- Step 2

  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to 1.

- Step 3

  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS2~SAINS0 bits.

  Select the external channel input to be converted, go to Step 4.

  Select the internal analog signal to be converted, go to Step 5.

- Step 4

  If the A/D input signal comes from the external channel input selected by configuring the SAINS2~SAINS0 bits, the corresponding pins should be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS3~SACS0 bits. After this step, go to Step 6.

- Step 5

  Before the A/D input signal is selected to come from the internal analog signal by configuring the SAINS2~SAINS0 bits, the corresponding external input pin must not be selected to an ANx pin by properly configured the pin-shared function selection bits. The desired internal analog signal then can be selected by configuring the SAINS2~SAINS0 bits. After this step, go to Step 6.

- Step 6

  Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register. Care should be taken in this step which can refer to the A/D Converter Reference Voltage section for details.

- Step 7

  Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.

- Step 8

  If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, the multi-function interrupt enable bit, MF0E, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.

- Step 9

  The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.

- Step 10

  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

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

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by clearing bit ADCEN to 0 in the

SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/O pins, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Conversion Function

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

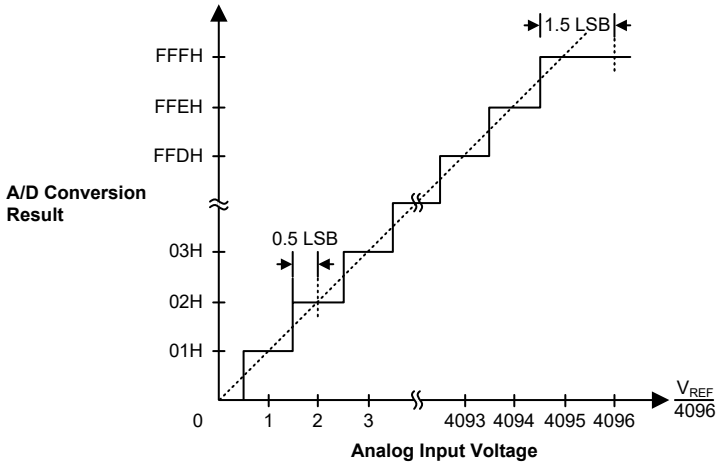
$$1\ LSB=V_{REF}\div 4096$$

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

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

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

Note that here the V_{REF} voltage is the actual A/D converter reference voltage determined by the SAVRS1~SAVRS0 bits.



Ideal A/D Transfer Function

Temperature Measurement Function

As the temperature sensor output voltage, V_{TSO}, has a linear relationship with temperature, the V_{TSO} A/D converted data value will also have a linear relationship with temperature. The current temperature Tx can be proportionally calculated from its A/D converted value ADCx using the following formula.

$$Tx=Remove\ LSB\ 12\ bits\ of\ (slope\times ADCx) - T_{OS};\ take\ lower\ 12\ bits\ of\ the\ final\ result$$

The slope and T_{OS} code are stored in the Option Memory and can be read from the Program Memory last page using the table read instruction when the Option Memory mapping function is enabled.

At the end of every conversion or say MCU calculation, the final data Tx is in two's complement format, has a data width of 12 bits and a resolution of 0.0625 (1/16)°C. The following table shows multiple examples of binary or hex data that can be read as temperature result.

Temperature	Digital Format (1 LSB=1/16=0.0625°C)	
Tx(°C)	BINARY	HEX
-40	1101 1000 0000	D80
-25	1110 0111 0000	E70
-10	1111 0110 0000	F60
-1	1111 1111 0000	FF0
-0.25	1111 1111 1100	FFC
-0.0625	1111 1111 1111	FFF
0	0000 0000 0000	000
0.0625	0000 0000 0001	001
0.25	0000 0000 0100	004
1	0000 0001 0000	010
10	0000 1010 0000	0A0
25	0001 1001 0000	190
40	0010 1000 0000	280
70	0100 0110 0000	460
85	0101 0101 0000	550

For example:

- Step 1

  Read the Option Memory slop=0x0AB9, T_{OS}=0x407.

- Step 2

  Enable temperature sensor & A/D function, then start A/D and read back A/D data registers.

- Step 3

  If ADCx=0x514, slope×ADCx=0x367374. Then remove the lower LSB 12 bits, get 0x0367.

- Step 4

  Minus T_{OS}, the result is 0x0367-0x407=FF60H (2's complement). Take the lower LSB 12 bits, then Tx=F60H=-10.0°C.

Name	Mapped Address in Program Memory	Description
slope	3FD5H	16-bit slope value bit 15~bit 8
	3FD6H	16-bit slope value bit 7~bit 0
T_{OS}	3FD7H	12-bit T_{OS} value bit 11~bit 4
	3FD8H	12-bit T_{OS} value bit 3~bit 0 (1LSB=1/16°C)

Temperature Measurement Reference Items

The Option Memory mapping function is enabled using the ORMC register. For more details, refer to the "Option Memory Mapping Register – ORMC" in the Special Function Register Description section.

A/D Conversion Programming Examples

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

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

```
clr  ADE                ; disable ADC interrupt
clr  TSEN               ; disable temperature sensor circuitry
clr  SAMS               ; set A/D converter conversion in the normal mode
mov  a, 0BH
mov  SADC1, a           ; select input signal from external channel input, reference
                        ; voltage from A/D internal power and fSYS/8 as A/D clock
mov  a, 02h             ; set PAS1 register to configure pin AN0
mov  PAS1, a
mov  a, 20h
mov  SADC0, a           ; enable A/D converter and connect AN0 channel to A/D
                        ; converter
:
start_conversion:
clr  START              ; high pulse on start bit to initiate conversion
set  START              ; reset A/D
clr  START              ; start A/D
polling_EOC:
sz   ADBZ               ; poll the SADC0 register ADBZ bit to detect end of A/D
                        ; conversion
jmp  polling_EOC        ; continue polling
mov  a, SADOL           ; read low byte conversion result value
mov  SADOL_buffer, a    ; save result to user defined register
mov  a, SADOH           ; read high byte conversion result value
mov  SADOH_buffer, a    ; save result to user defined register
:
:
jmp  start_conversion   ; start next A/D conversion
```

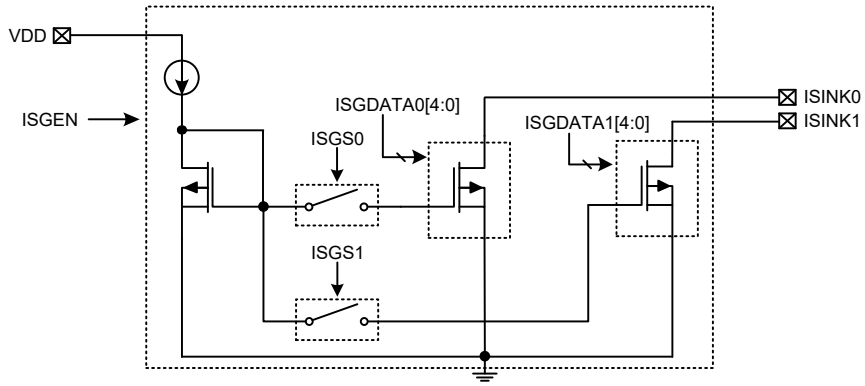
Example: using the interrupt method to detect the end of conversion

```
clr  ADE                ; disable ADC interrupt
clr  TSEN               ; disable temperature sensor circuitry
clr  SAMS               ; set A/D converter conversion in the normal mode
mov  a, 0BH
mov  SADC1, a           ; select input signal from external channel input, reference
                        ; voltage from A/D internal power and fSYS/8 as A/D clock
mov  a, 02h             ; set PAS1 register to configure pin AN0
mov  PAS1, a
mov  a, 20h
mov  SADC0, a           ; enable A/D converter and connect AN0 channel to A/D
                        ; converter
Start_conversion:
clr  START              ; high pulse on START bit to initiate conversion
set  START              ; reset A/D
clr  START              ; start A/D
clr  ADF                ; clear ADC interrupt request flag
set  ADE                ; enable ADC interrupt
set  EMI                ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov  acc_stack, a       ; save ACC to user defined memory
mov  a, STATUS
mov  status_stack, a    ; save STATUS to user defined memory
:
:
```

```
mov  a, SADOL            ; read low byte conversion result value
mov  SADOL_buffer, a     ; save result to user defined register
mov  a, SADOH            ; read high byte conversion result value
mov  SADOH_buffer, a     ; save result to user defined register
:
:
EXIT_INT_ISR:
mov  a, status_stack
mov  STATUS, a           ; restore STATUS from user defined memory
mov  a, acc_stack        ; restore ACC from user defined memory
reti
```

Sink Current Generator

The sink current source generator could provide constant current no matter what V_{ISINK} voltage is from 0.7V to 4.5V, and the V_{DD} voltage is from 2.2V to 5.5V. The constant current value is controlled by the ISGDATA0/ISGDATA1 register, and the sink current range is 50mA~360mA.



Sink Current Generator Registers

There are a series of registers which control the overall operation of the Sink Current Generator function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
ISGENC	ISGEN	—	—	—	—	—	ISGS1	ISGS0
ISGDATA0	—	—	—	D4	D3	D2	D1	D0
ISGDATA1	—	—	—	D4	D3	D2	D1	D0

Sink Current Generator Register List

• ISGENC Register

Bit	7	6	5	4	3	2	1	0
Name	ISGEN	—	—	—	—	—	ISGS1	ISGS0
R/W	R/W	—	—	—	—	—	R/W	R/W
POR	0	—	—	—	—	—	0	0

Bit 7 **ISGEN**: Sink current generator enable control
0: Disable
1: Enable
When the ISGEN bit is cleared to zero to disable the sink current generator, the ISINK0 and ISINK1 pin status are $V_{ISINK0/1}$=floating, $I_{ISINK0/1}$=0.

Bit 6~2 Unimplemented, read as "0"

Bit 1 **ISGS1**: ISINK1 pin sink current control
0: Disable
1: Enable

Bit 0 **ISGS0**: ISINK0 pin sink current control
0: Disable
1: Enable

• ISGDATA0 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**: Sink current generator control for ISINK0 pin
Current value (mA)=50+10×(ISGDATA0[4:0])
When the ISGS0 bit is cleared to 0, the ISGDATA0[4:0] should be cleared to 00000 to avoid the leakage situation occurrence.
Refer to "Sink Current Generator Electrical Characteristics" table for more details.

• ISGDATA1 Register

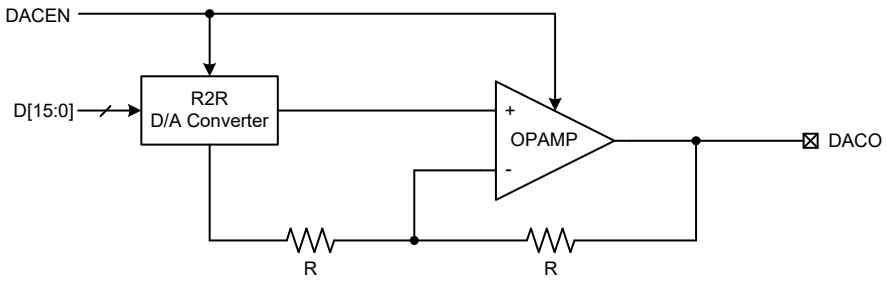
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	D4	D3	D2	D1	D0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **D4~D0**: Sink current generator control for ISINK1 pin
Current value (mA)=50+5×(ISGDATA1[4:0])
When the ISGS1 bit is cleared to 0, the ISGDATA1[4:0] should be cleared to 00000 to avoid the leakage situation occurrence.
Refer to "Sink Current Generator Electrical Characteristics" table for more details.

16-bit Voice D/A Converter

The device contains a 16-bit D/A Converter. The circuit is a 16-bit R2R D/A Converter for audio application. Its reference voltage comes from analog supply voltage only, and can be powered down to save power. The 16-bit D/A Converter is suitable for voice or audio application. Although this D/A Converter is not a general one-to-one digital to analog converter, it provides not bad and same audio quality no matter the voice signal is small or large. Note that the D/A Converter voltage is amplified and buffer output by OPAMP.



16-bit D/A Converter Block Diagram

16-bit Voice D/A Converter Registers

Overall operation of the D/A Converter is controlled by using three registers. There is a 16-bit D/A Converter data high byte register, DAH, a 16-bit D/A Converter data low byte register, DAL, and a control register named as DACC used to control the D/A converter function enable or disable.

Register Name	Bit							
	7	6	5	4	3	2	1	0
DAH	D15	D14	D13	D12	D11	D10	D9	D8
DAL	D7	D6	D5	D4	D3	D2	D1	D0
DACC	—	—	—	—	—	—	—	DACEN

16-bit D/A Converter Register List

• DAH Register

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

Bit 7~0 **D15~D8**: 16-bit voice D/A converter data high byte

The 16-bit D/A converter Data low byte register, known as DAL, should first be modified and then followed by the DAH register modification. Each time when the DAH register is written, the whole 16-bit data will be loaded into the D/A converter and a conversion cycle will be initiated. Note that the D/A converter should first be enabled before the D/A converter data is updated.

• DAL register

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

Bit 7~0 **D7~D0**: 16-bit voice D/A converter data low byte

Writing this register will only write the data to the shadow buffer and writing the DAH register will simultaneously copy the shadow buffer data to the DAL register. Note that the D/A converter should first be enabled before the D/A converter is updated.

• DACC register

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **DACEN**: 16-bit voice D/A converter enable or disable control

0: Disable

1: Enable

If the D/A converter is enabled, users must wait a t_{DACS} time to ensure the D/A converter circuit is stable. And the D/A converter data register should be updated after the D/A converter circuit is stable.

Serial Interface Module – SIM

The device contains a Serial Interface Module, which includes both the four-line SPI interface or two-line I²C 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 I²C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins and therefore the SIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As both interface types share the same pins and registers, the choice of whether the SPI or I²C type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O pins are selected using pull-high control registers when the SIM function is enabled and the corresponding pins are used as SIM input pins.

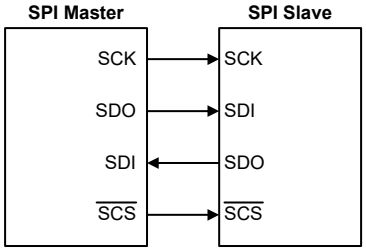
SPI Interface

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

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

SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin 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 pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ 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.



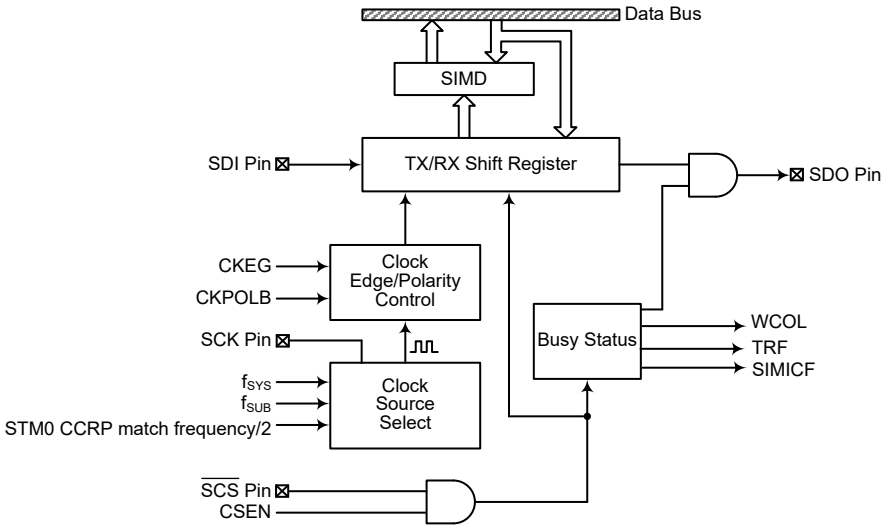
SPI Master/Slave Connection

The SPI function in the device offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes
- LSB first or MSB first data transmission modes

- Transmission complete flag
- Rising or falling active clock edge

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



SPI Block Diagram

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two control registers, SIMC0 and SIMC2.

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

SPI Register List

SPI Data Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and 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

Bit 7~0 **D7~D0**: SIM data register bit 7 ~ bit 0

SPI Control Registers

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I²C function. The SIMC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC2 register is used for other control functions such as LSB/MSB selection, write collision flag, etc.

- **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	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 STM0 CCRP interrupt frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused

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 STM0 CCRP interrupt and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection

These bits are only available when the SIM is configured to operate in the I²C mode. Refer to the I²C register section.

Bit 1 **SIMEN**: SIM Enable Control
0: Disable
1: Enable

The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an 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 conmunication incompleted did not occur
1: SIM SPI conmunication incompleted occurred

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

• **SIMC2 Register**

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

Bit 7~6 **D7~D6**: Undefined bits

These bits can be read or written by application program.

Bit 5 **CKPOLB**: SPI clock line base condition selection

0: The SCK line will be high when the clock is inactive

1: The SCK line will be low when the clock is inactive

The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4 **CKEG**: SPI SCK clock active edge type selection

CKPOLB=0

0: SCK is high base level when the clock is inactive and data capture at SCK rising edge

1: SCK is high base level when the clock is inactive and data capture at SCK falling edge

CKPOLB=1

0: SCK is low base level when the clock is inactive and data capture at SCK falling edge

1: SCK is low base level when the clock is inactive and data capture at SCK rising edge

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

Bit 3 **MLS**: SPI data shift order

0: LSB first

1: MSB first

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

Bit 2 **CSEN**: SPI $\overline{\text{SCS}}$ pin control

0: Disable

1: Enable

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

Bit 1 **WCOL**: SPI write collision flag

0: No collision

1: Collision

The WCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program.

Bit 0 **TRF**: SPI Transmit/Receive complete flag

0: SPI data is being transferred

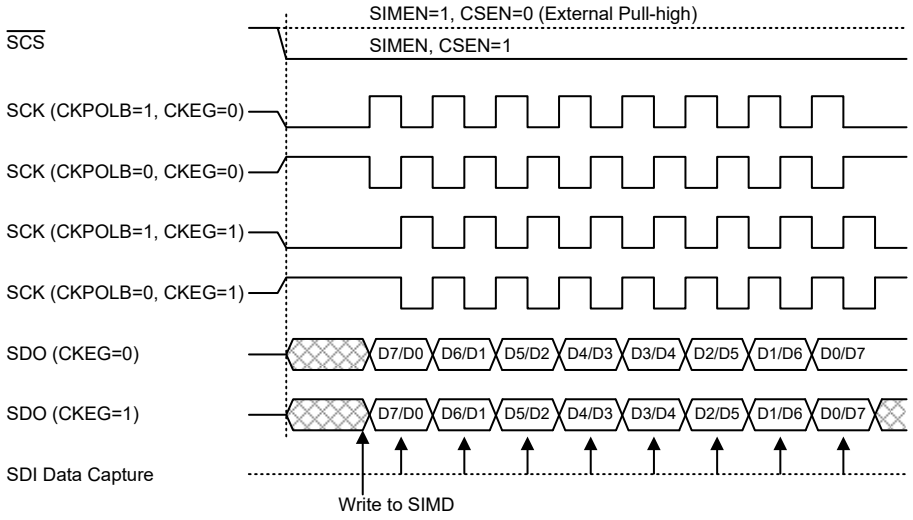
1: SPI data transmission is completed

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

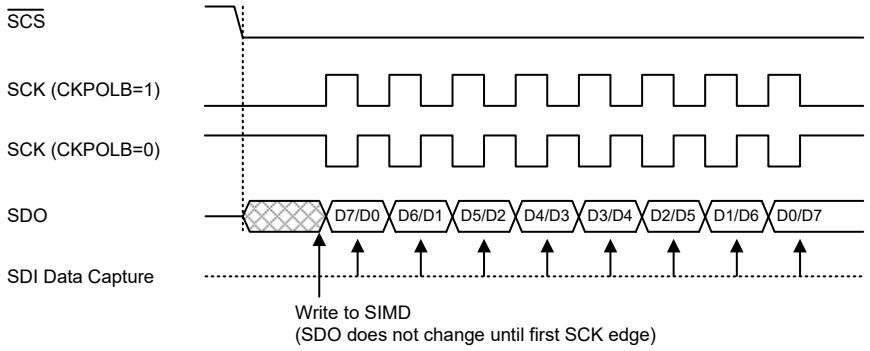
SPI Communication

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

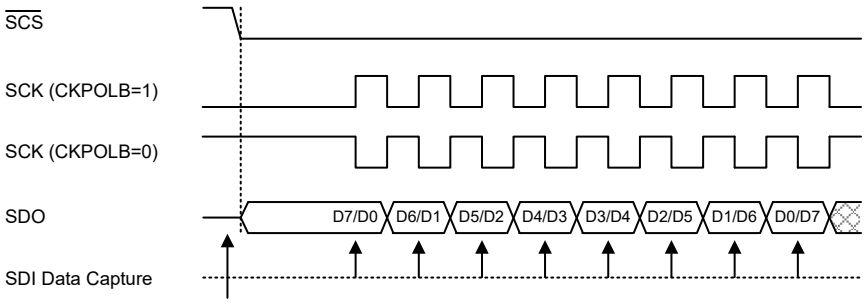
The SPI Master mode will continue to function if the SPI clock is running.



SPI Master Mode Timing

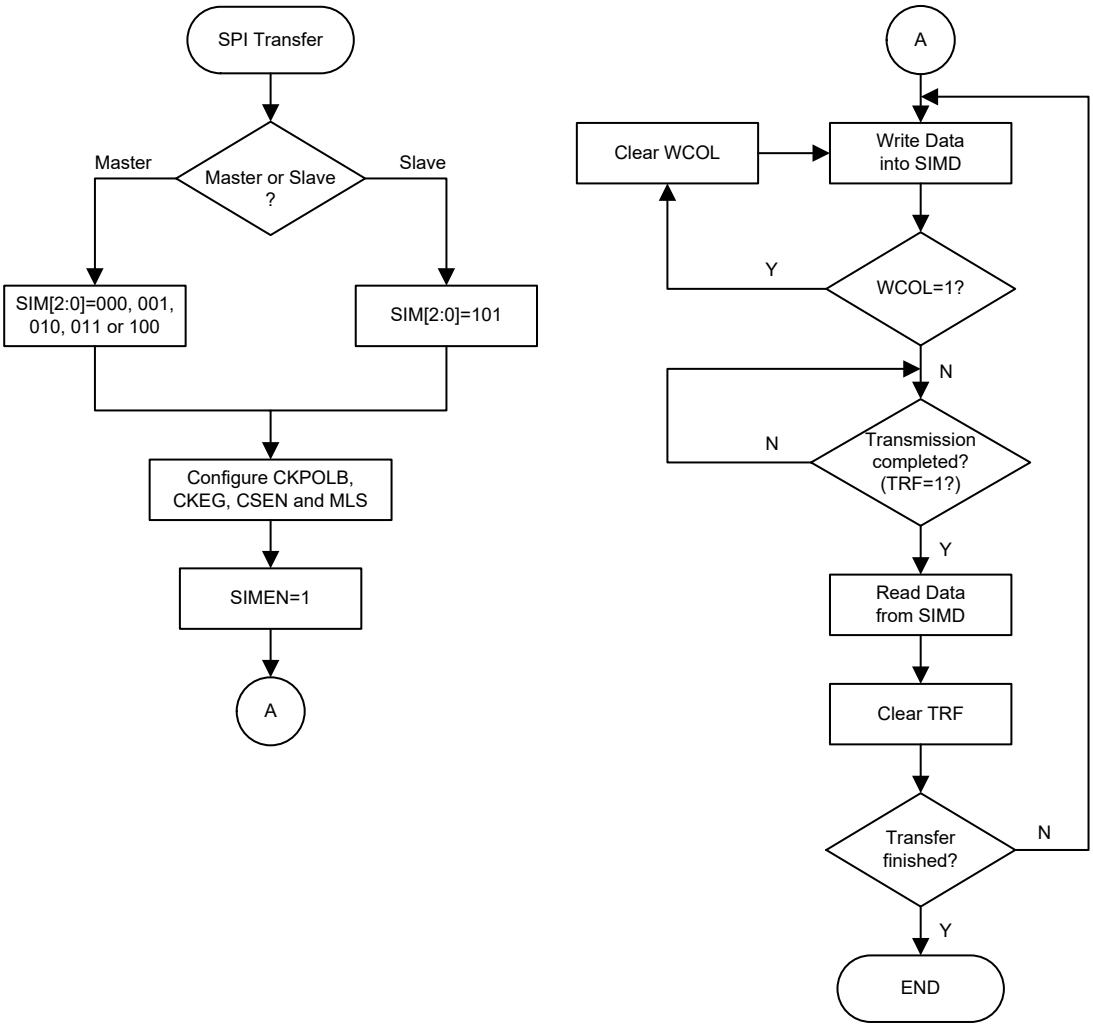


SPI Slave Mode Timing – CKEG=0



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

SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flowchart

SPI Bus Enable/Disable

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

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

SPI Operation Steps

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

The CSEN bit in the SIMC2 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{SCS}$ line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the $\overline{SCS}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the SDI line in a floating condition and the SDO line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and the $\overline{SCS}$, SDI, SDO and SCK will all become I/O pins or the other functions using the corresponding pin-shared control bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SIMD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode:

- Step 1

  Select the SPI Master mode and clock source using the SIM2~SIM0 bits in the SIMC0 control register.

- Step 2

  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave devices.

- Step 3

  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.

- Step 4

  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then use the SCK and SDO lines to output the data. After this, go to step5.

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

- Step 5

  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.

- Step 6

  Check the TRF bit or wait for a SIM SPI serial bus interrupt.

- Step 7

  Read data from the SIMD register.

- Step 8

  Clear TRF.

- Step 9

  Go to step 4.

Slave Mode:

- Step 1

  Select the SPI Slave mode using the SIM2~SIM0 bits in the SIMC0 control register.

- Step 2

  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.

- Step 3

  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.

- Step 4

  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and $\overline{SCS}$ signal. After this, go to step5.

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

- Step 5

  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.

- Step 6

  Check the TRF bit or wait for a SIM SPI serial bus interrupt.

- Step 7

  Read data from the SIMD register.

- Step 8

  Clear TRF.

- Step 9

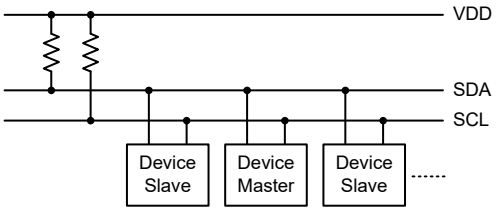
  Go to step 4.

Error Detection

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

I²C Interface

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

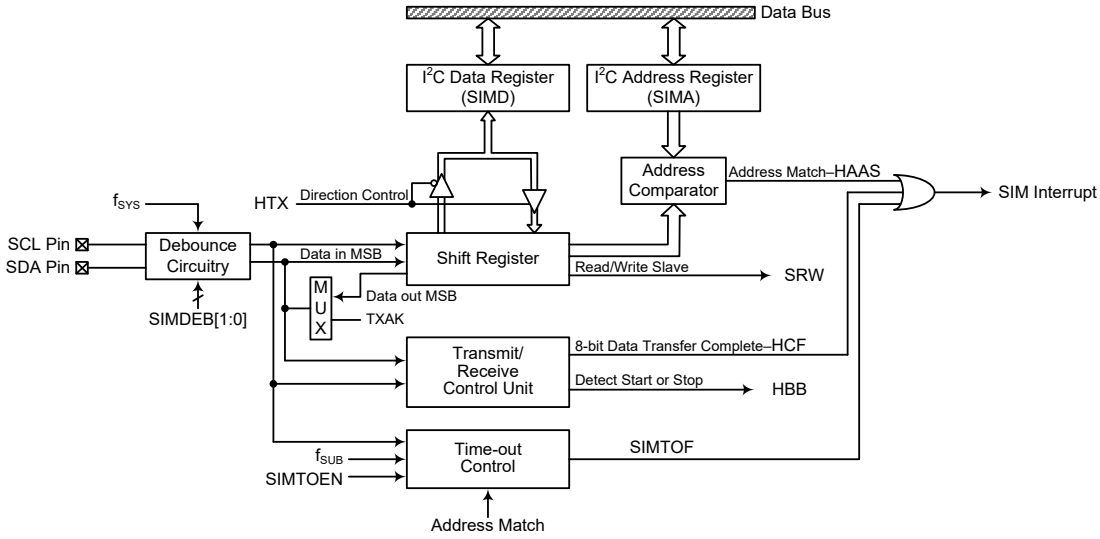


I²C Master Slave Bus Connection

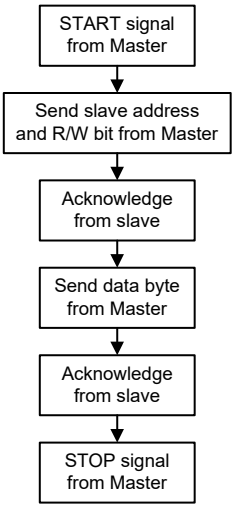
I²C Interface Operation

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 device, which only operates 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-high control function pin-shared with SCL/SDA pin is still applicable even if I²C device is activated and the related internal pull-high function could be controlled by its corresponding pull-high control register.



I²C Block Diagram



I²C Interface Operation

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

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

I²C Minimum f_{SYS} Frequency Requirements

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.

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	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C Register List

I²C Data Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C function. Before the device writes 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

Bit 7~0 **D7~D0**: SIM data register bit 7 ~ bit 0

I²C Address Register

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

• **SIMA Register**

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

Bit 7~1 **SIMA6~SIMA0**: I²C slave address
SIMA6~SIMA0 is the I²C slave address bit 6 ~ bit 0.

Bit 0 **D0**: Reserved, can be read or written

I²C Control Registers

There are three control registers for the I²C interface, SIMC0, SIMC1 and SIMTOC. The SIMC0 register is used to control the enable/disable function and to select the I²C slave mode and debounce time. The SIMC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, SIMTOC, is used to control the I²C time-out function and is described in the corresponding section.

• **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is STM0 CCRP interrupt frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused

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 STM0 CCRP interrupt and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
1x: 4 system clock debounce
These bits are used to select the I²C debounce time when the SIM is configured as the I²C interface function by setting the SIM2~SIM0 bits to "110".

Bit 1 **SIMEN**: SIM Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an 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
This bit is only available when the SIM is configured to operate in an SPI slave mode.
Refer to the SPI register section.

- **SIMC1 Register**

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

Bit 7 **HCF**: 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 zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated. Below is an example of the flow of a two-byte I²C data transfer.
First, the I²C slave device receives a start signal from I²C master and then HCF bit is automatically cleared to zero. Second, the I²C slave device finishes receiving the 1st data byte and then the HCF bit is automatically set high. Third, users read the 1st data byte from SIMD register by the application program and then the HCF bit is automatically cleared to zero. Fourth, the I²C slave device finishes receiving the 2nd data byte and then HCF bit is automatically set high and so on. Finally, the I²C slave device receives a stop signal from the I²C master and then the HCF bit is automatically set high.

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 set 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 bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set TXAK bit to "0" before further data is received.

Bit 2 **SRW**: 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 zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

Bit 1 **IAMWU**: I²C Address Match Wake-up MCU control
0: Disable
1: Enable

This bit should be set to 1 to enable the I²C address match wake up MCU from the SLEEP or IDLE Mode. If the IAMWU bit has been set high before entering either the SLEEP or IDLE mode to enable the I²C address match wake up MCU, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

Bit 0 **RXAK**: I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave does not receive acknowledge flag

The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

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 a SIM I²C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer completion or from the I²C bus time-out occurrence. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave

device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1

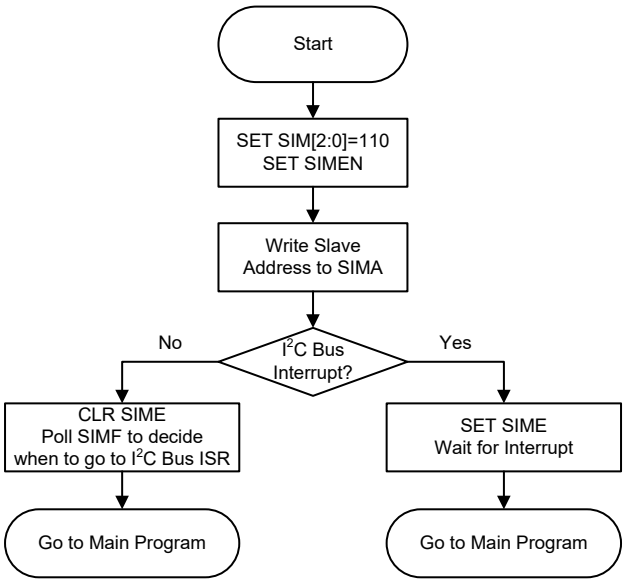
  Set the SIM2~SIM0 and SIMEN bits in the SIMC0 register to "110" and "1" respectively to enable the I²C bus.

- Step 2

  Write the slave address of the device to the I²C bus address register SIMA.

- Step 3

  Set the interrupt enable bit SIME of the interrupt control register to enable the SIM interrupt.



I²C Bus Initialisation Flow Chart

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.

I²C 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 a SIM I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits should be examined to see whether the interrupt source

has come from a matching slave address or from the completion of a data byte transfer or the I²C bus time-out occurrence. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

I²C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the master 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.

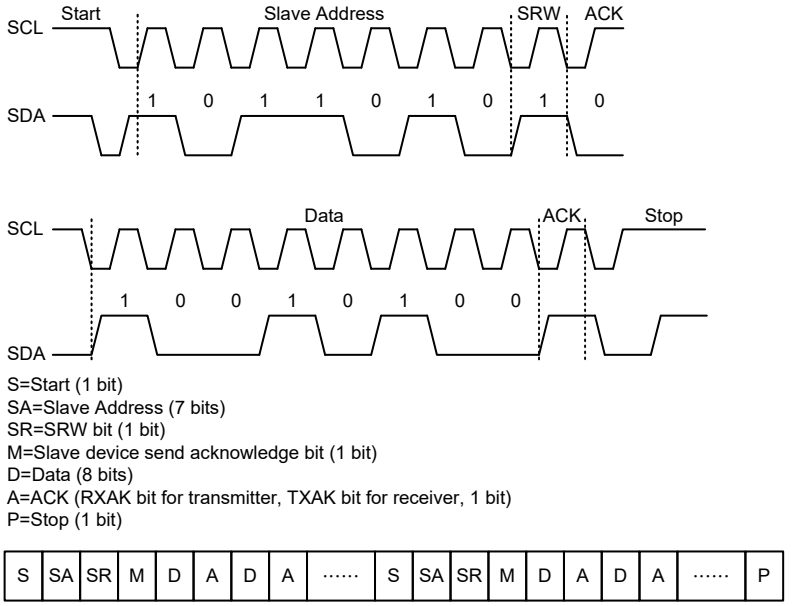
I²C Bus Slave Address Acknowledge Signal

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 to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be set to "0".

I²C Bus Data and Acknowledge Signal

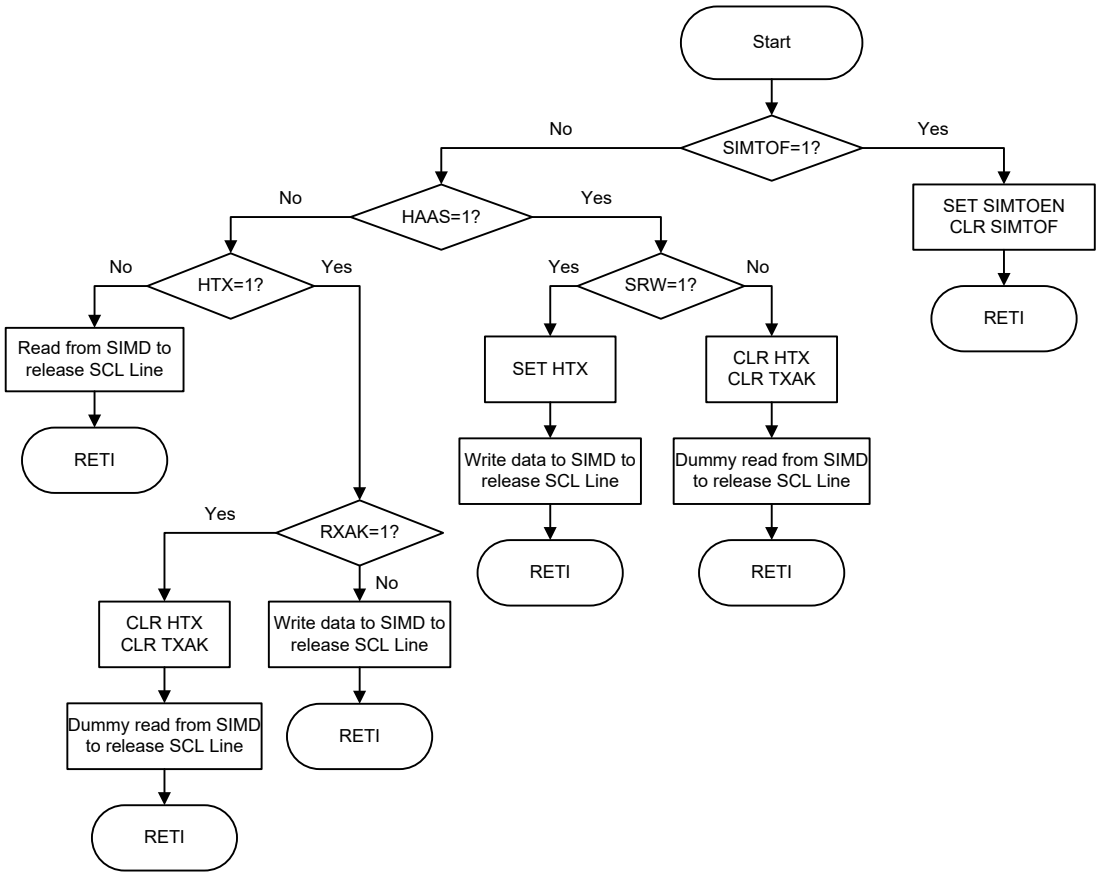
The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level "0", before it can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



I²C Communication Timing Diagram

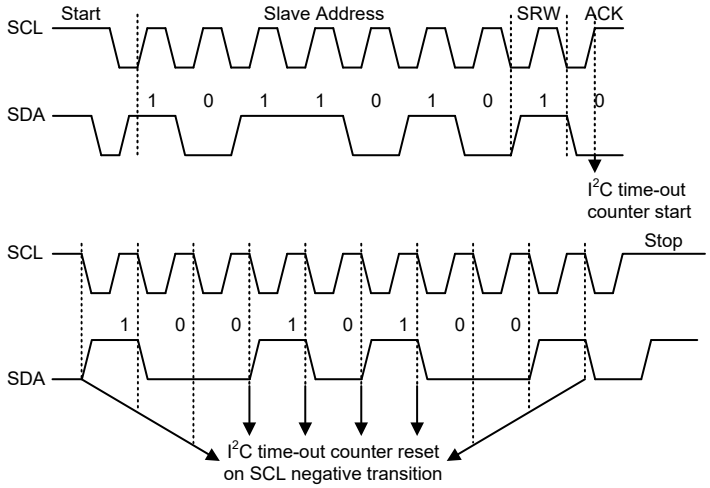
Note: *When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.



I²C Bus ISR Flow Chart

I²C Time-out Control

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



I²C Time-out Function

When an I²C time-out counter overflow occurs, the counter will stop and the SIMTOEN bit will be cleared to zero and the SIMTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the I²C interrupt vector. 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

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

- **SIMTOC Register**

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

Bit 7 **SIMTOEN**: SIM I²C Time-out control
0: Disable
1: Enable

Bit 6 **SIMTOF**: SIM I²C Time-out flag
0: No time-out occurred
1: Time-out occurred

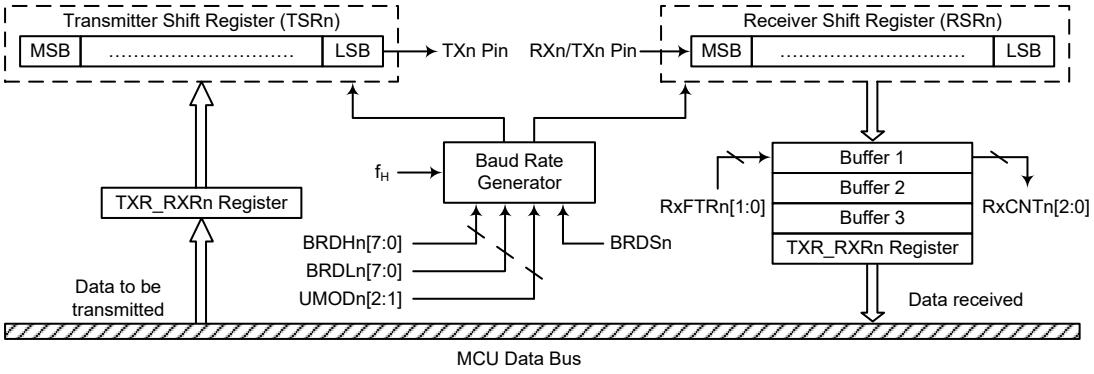
Bit 5~0 **SIMTOS5~SIMTOS0**: SIM I²C Time-out period selection
I²C time-out clock source is $f_{SUB}/32$.
I²C time-out period=(SIMTOS[5:0]+1)×($32/f_{SUB}$).

UART Interfaces

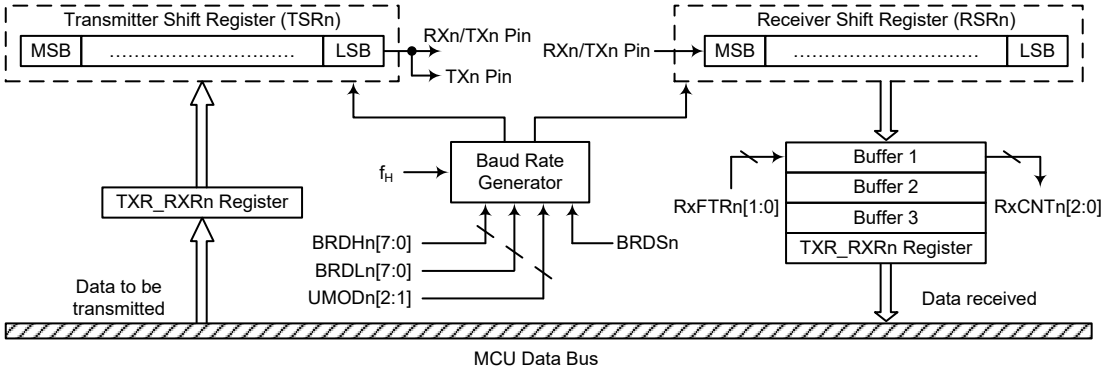
The device contains two integrated full-duplex or half-duplex asynchronous serial communications UART interfaces that enable communication with external devices that contain a serial interface. Each UART function has many features and can transmit and receive data serially by transferring a frame of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

Each integrated UART function contains the following features:

- Full-duplex or half-duplex (single wire mode), asynchronous communication
- 8 or 9 bits character length
- Even, odd, mark, space or no parity options
- One or two stop bits configurable for receiver
- Two stop bits for transmitter
- Baud rate generator with 16-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 4-byte Deep FIFO Receive Data Buffer
- 1-byte Deep FIFO Transmit Data Buffer
- RXn/TXn pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
  - Transmitter Empty
  - Transmitter Idle
  - Receiver reaching FIFO trigger level
  - Receiver Overrun
  - Address Mode Detect



UARTn Data Transfer Block Diagram – SWMn=0 (n=0~1)



UARTn Data Transfer Block Diagram – SWMn=1 (n=0~1)

UART External Pins

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

UART Single Wire Mode

The UARTn function also supports a Single Wire Mode communication which is selected using the SWMn bit in the UnCR3 register. When the SWMn bit is set high, the UARTn function will be in the single wire mode. In the single wire mode, a single RXn/TXn pin can be used to transmit and receive data depending upon the corresponding control bits. When the RXENn bit is set high, the RXn/TXn pin is used as a receiver pin. When the RXENn bit is cleared to zero and the TXENn bit is set high, the RXn/TXn pin will act as a transmitter pin.

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

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

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

UART Data Transfer Scheme

The UARTn Data Transfer Block Diagram shows the overall data transfer structure arrangement for the UARTn. The actual data to be transmitted from the MCU is first transferred to the TXR_RXRn register by the application program. The data will then be transferred to the Transmit Shift Register

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

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

UART Status and Control Registers

There are nine control registers associated with the UARTn function. The SWMn bit in the UnCR3 register is used to enable/disable the UARTn Single Wire Mode.The UnSR, UnCR1, UnCR2, UFCRn and RxCNTn registers control the overall function of the UARTn, while the BRDHn and BRDLn register control the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXRn data register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
UnSR	PERRn	NFn	FERRn	OERRn	RIDLEn	RXIFn	TIDLEn	TXIFn
UnCR1	UARTENn	BNOn	PRENn	PRTn1	PRTn0	TXBRKn	RX8n	TX8n
UnCR2	TXENn	RXENn	STOPSn	ADDENn	WAKEn	RIEn	TIIEn	TEIEn
UnCR3	—	—	—	—	—	—	—	SWMn
TXR_RXRn	D7	D6	D5	D4	D3	D2	D1	D0
BRDHn	D7	D6	D5	D4	D3	D2	D1	D0
BRDLn	D7	D6	D5	D4	D3	D2	D1	D0
UFCRn	—	—	UMODn2	UMODn1	UMODn0	BRDSn	RxFTRn1	RxFTRn0
RxCNTn	—	—	—	—	—	D2	D1	D0

UARTn Register List(n=0~1)

• UnSR Register

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

Bit	7	6	5	4	3	2	1	0
Name	PERRn	NFn	FERRn	OERRn	RIDLEn	RXIFn	TIDLEn	TXIFn
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

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

The PERRn flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if the parity is enabled and the parity type (odd, even, mark or space) is selected. The flag can also be cleared to zero by a software sequence which involves a read to the status register UnSR followed by an access to the TXR_RXRn data register.

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

The NFn flag is the noise flag. When this read only flag is "0", it indicates no noise condition. When the flag is "1", it indicates that the UARTn has detected noise on the receiver input. The NFn flag is set during the same cycle as the RXIFn flag but will not be set in the case of as overrun. The NFn flag can be cleared to zero by a software sequence which will involve a read to the status register UnSR followed by an access to the TXR_RXRn data register.

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

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

Bit 4 **OERRn**: Overrun error flag
0: No overrun error is detected
1: Overrun error is detected

The OERRn flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is "0", it indicates that there is no overrun error. When the flag is "1", it indicates that an overrun error occurs which will inhibit further transfers to the TXR_RXRn receive data register. The flag is cleared to zero by a software sequence, which is a read to the status register UnSR followed by an access to the TXR_RXRn data register.

Bit 3 **RIDLEn**: Receiver status
0: Data reception is in progress (Data being received)
1: No data reception is in progress (Receiver is idle)

The RIDLEn flag is the receiver status flag. When this read only flag is "0", it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is "1", it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLEn bit is "1" indicating that the UARTn receiver is idle and the RXn/TXn pin stays in logic high condition.

Bit 2 **RXIFn**: Receive TXR_RXRn data register status
0: TXR_RXRn data register is empty
1: TXR_RXRn data register has available data and Receiver FIFO trigger level is reached

The RXIFn flag is the receive data register status flag. When this read only flag is "0", it indicates that the TXR_RXRn read data register is empty. When the flag is "1", it indicates that the TXR_RXRn read data register contains new data and Receiver FIFO trigger level is reached. When the contents of the shift register are transferred to the TXR_RXRn register and Receiver FIFO trigger level is reached, an interrupt is generated if RIEn=1 in the UnCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NFn, FERRn, and/or PERRn are set within the same clock cycle. The RXIFn flag will eventually be cleared to zero when the UnSR register is read with RXIFn set, followed by a read from the TXR_RXRn register, and if the TXR_RXRn register has no more new data available.

Bit 1 **TIDLEn**: Transmission idle
0: Data transmission is in progress (Data being transmitted)
1: No data transmission is in progress (Transmitter is idle)

The TIDLEn flag is known as the transmission complete flag. When this read only flag is "0", it indicates that a transmission is in progress. This flag will be set high when the TXIFn flag is "1" and when there is no transmit data or break character being transmitted. When TIDLEn is equal to "1", the TXn pin becomes idle with the pin

state in logic high condition. The TIDLEn flag is cleared to zero by reading the UnSR register with TIDLEn set and then writing to the TXR_RXRn register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 **TXIFn**: Transmit TXR_RXRn data register status
0: Character is not transferred to the transmit shift register
1: Character has transferred to the transmit shift register (TXR_RXRn data register is empty)

The TXIFn 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_RXRn data register. The TXIFn flag is cleared to zero by reading the UARTn status register (UnSR) with TXIFn set and then writing to the TXR_RXRn data register. Note that when the TXENn bit is set, the TXIFn flag bit will also be set since the transmit data register is not yet full.

• UnCR1 Register

The UnCR1 register together with the UnCR2 and UnCR3 registers are the three UARTn control registers that are used to set the various options for the UARTn function, such as overall on/off control, parity control, data transfer bit length, single wire mode communication etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UARTENn	BNOn	PRENn	PRTn1	PRTn0	TXBRKn	RX8n	TX8n
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 **UARTENn**: UARTn function enable control
0: Disable UARTn. TXn and RXn/TXn pins are in a floating state
1: Enable UARTn. TXn and RXn/TXn pins can function as UARTn pins

The UARTENn bit is the UARTn enable bit. When this bit is equal to "0", the UARTn will be disabled and the RXn/TXn pin as well as the TXn pin will be set in a floating state. When the bit is equal to "1", the UARTn will be enabled and the TXn and RXn/TXn pins will function as defined by the SWM mode selection bit together with the TXENn and RXENn enable control bits.

When the UARTn 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 UARTn is disabled, all error and status flags will be reset. Also the TXENn, RXENn, TXBRKn, RXIFn, OERRn, FERRn, PERRn and NFn bits as well as the RxCNT register will be cleared to zero, while the TIDLEn, TXIFn and RIDLEn bits will be set high. Other control bits in UnCR1, UnCR2, UnCR3, BRDHn and BRDLn registers will remain unaffected. If the UARTn is active and the UARTENn bit is cleared to zero, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UARTn is re-enabled, it will restart in the same configuration.

Bit 6 **BNOn**: 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 RX8n and TX8n will be used to store the 9th bit of the received and transmitted data respectively.

Note that the 9th bit of data if BNOn=1, or the 8th bit of data if BNOn=0, which is used as the parity bit, does not transfer to RX8n or TXRX7n respectively when the parity function is enabled.

Bit 5 **PRENn**: 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 4~3 **PRTn1~ PRTn0**: Parity type selection bit
00: Even parity for parity generator
01: Odd parity for parity generator
10: Mark parity for parity generator
11: Space parity for parity generator
These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.

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

Bit 1 **RX8n**: Receive data bit 8 for 9-bit data transfer format (read only)
This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8n. The BNOn bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **TX8n**: Transmit data bit 8 for 9-bit data transfer format (write only)
This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8n. The BNOn bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• UnCR2 Register

The UnCR2 register is the second of the three UARTn control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UARTn Transmitter and Receiver as well as enabling the various UARTn interrupt sources. The register also serves to control the receiver STOP bit number selection, receiver wake-up function enable and the address detect function enable. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	TXENn	RXENn	STOPSn	ADDENn	WAKEn	RIEn	TIIEn	TEIEn
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

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

Bit 6 **RXENn**: UARTn Receiver enabled control
0: UARTn receiver is disabled
1: UARTn receiver is enabled

The bit named RXENn is the Receiver Enable Bit. When this bit is equal to "0", the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RXn/TXn pin will be set in a floating state.

If the RXENn bit is equal to "1" and the UARTENn bit is also equal to "1", the receiver will be enabled and the RXn pin will be controlled by the UARTn. Clearing the RXENn bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RXn/TXn pin will be set in a floating state.

Bit 5 **STOPSn**: Number of stop bits selection for receiver
0: One stop bit format is used
1: Two stop bits format is used

This bit determines if one or two stop bits are to be used for receiver. When this bit is equal to "1", two stop bits format are used. If the bit is equal to "0", then only one stop bit format is used. Two stop bits are used for transmitter.

Bit 4 **ADDENn**: Address detect function enable control
0: Address detect function is disabled
1: Address detect function is enabled

The bit named ADDENn is the address detect function enable control bit. When this bit is equal to "1", the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXRX7 if BNOn=0 or the 9th bit, which corresponds to RX8n if BNOn=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNOn. If the address bit known as the 8th or 9th bit of the received word is "0" with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **WAKEn**: RXn/TXn pin wake-up UARTn function enable control
0: RXn/TXn pin wake-up UARTn function is disabled
1: RXn/TXn pin wake-up UARTn function is enabled

This bit is used to control the wake-up UARTn function when a falling edge on the RXn/TXn pin occurs. Note that this bit is only available when the UARTn clock (f_H) is switched off. There will be no RXn/TXn pin wake-up UARTn function if the UARTn clock (f_H) exists. If the WAKEn bit is set to 1 as the UARTn clock (f_H) is switched off, a UARTn wake-up request will be initiated when a falling edge on the RXn/TXn pin occurs. When this request happens and the corresponding interrupt is enabled, an RXn/TXn pin wake-up UARTn interrupt will be generated to inform the MCU to wake up the UARTn function by switching on the UARTn clock (f_H) via the application program. Otherwise, the UARTn function cannot resume even if there is a falling edge on the RXn/TXn pin when the WAKEn bit is cleared to 0.

Bit 2 **RIEn**: Receiver interrupt enable control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled

This bit enables or disables the receiver interrupt. If this bit is equal to "1" and when the receiver overrun flag OERRn or receive data available flag RXIFn is set, the UARTn interrupt request flag will be set. If this bit is equal to "0", the UARTn interrupt request flag will not be influenced by the condition of the OERRn or RXIFn flags.

Bit 1 **TIIEn**: Transmitter Idle interrupt enable control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled

This bit enables or disables the transmitter idle interrupt. If this bit is equal to "1" and when the transmitter idle flag TIDLEn is set, due to a transmitter idle condition, the UARTn interrupt request flag will be set. If this bit is equal to "0", the UARTn interrupt request flag will not be influenced by the condition of the TIDLEn flag.

Bit 0 **TEIEn**: Transmitter Empty interrupt enable control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled
This bit enables or disables the transmitter empty interrupt. If this bit is equal to "1" and when the transmitter empty flag TXIFn is set, due to a transmitter empty condition, the UARTn interrupt request flag will be set. If this bit is equal to "0", the UARTn interrupt request flag will not be influenced by the condition of the TXIFn flag.

• UnCR3 Register

The UnCR3 register is used to enable the UARTn Single Wire Mode communication. As the name suggests in the single wire mode the UARTn communication can be implemented in one single line, RXn/TXn, together with the control of the RXENn and TXENn bits in the UnCR2 register.

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **SWMn**: Single Wire Mode enable control
0: Disable, the RXn/TXn pin is used as UARTn receiver function only
1: Enable, the RXn/TXn pin can be used as UARTn receiver or transmitter function controlled by the RXENn and TXENn bits
Note that when the Single Wire Mode is enabled, if both the RXENn and TXENn bits are high, the RXn/TXn pin will just be used as UARTn receiver input.

• TXR_RXRn Register

The TXR_RXRn register is the data register which is used to store the data to be transmitted on the TXn pin or being received from the RXn/TXn pin.

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

"x": unknown

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

• BRDHn Register

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

Bit 7~0 **D7~D0**: Baud rate divider high byte
The baud rate divider BRD (BRDHn/BRDLn) defines the UARTn clock divider ratio.
Baud Rate=f_H/(BRD+UMODn/8)
BRD=16~65535 or 8~65535 depending on BRDSn
Note: 1. BRD value should not be set to less than 16 when BRDSn=0 or less than 8 when BRDSn=1, otherwise errors may occur.
2. The BRDLn must be written first and then BRDHn, otherwise errors may occur.
3. The BRDHn register should not be modified during data transmission process.

• **BRDLn Register**

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

"x": unknown

Bit 7~0 **D7~D0**: Baud rate divider low byte

The baud rate divider BRD (BRDHn/BRDLn) defines the UARTn clock divider ratio.

Baud Rate=f_H/(BRD+UMODn/8)

BRD=16~65535 or 8~65535 depending on BRDSn

Note: 1. BRD value should not be set to less than 16 when BRDSn=0 or less than 8 when BRDSn=1, otherwise errors may occur.

2. The BRDLn must be written first and then BRDHn, otherwise errors may occur.

3. The BRDLn register should not be modified during data transmission process.

• **UFCRn Register**

The UFCRn register is the FIFO control register which is used for UARTn modulation control, BRDn range selection and trigger level selection for RXIFn and interrupt.

Bit	7	6	5	4	3	2	1	0
Name	—	—	UMODn2	UMODn1	UMODn0	BRDSn	RxFTRn1	RxFTRn0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **UMODn2~UMODn0**: UARTn Modulation Control bits

The modulation control bits are used to correct the baud rate of the received or transmitted UARTn signal. These bits determine if the extra UARTn clock cycle should be added in a UARTn bit time. The UMODn2~ The UMODn0 will be added to internal accumulator for every UARTn bit time. Until a carry to bit 3, the corresponding UARTn bit time increases a UARTn clock cycle.

Bit 2 **BRDSn**: BRDn range selection

0: BRDn range is from 16 to 65535

1: BRDn range is from 8 to 65535

The BRDSn is used to control the sampling point in a UARTn bit time. If BRDSn is cleared to zero, the sampling point will be BRDn/2 , BRDn/2+1×F_h, and BRDn/2+2×F_h in a UARTn bit time. If the BRDSn is set high, the sampling point will be BRDn/2-1×F_h, BRDn/2, and BRDn/2+2×F_h in a UARTn bit time.

Note that the BRDSn bit should not be modified during data transmission process.

Bit 1~0 **RxFTRn1~ RxFTRn0**: Receiver FIFO trigger level (bytes)

00: 4 bytes in Receiver FIFO

01: 1 or more bytes in Receiver FIFO

10: 2 or more bytes in Receiver FIFO

11: 3 or more bytes in Receiver FIFO

For the receiver these bits define the number of received data bytes in the Receiver FIFO that will trigger the RXIFn bit being set high, an interrupt will also be generated if the RIEn bit is enabled. To prevent OERRn from being set high, the receiver FIFO trigger level can be set to 2 bytes, avoiding an overrun state that cannot be processed by the program in time when more than 4 data bytes are received. After the reset the receiver FIFO is empty.

• **RxCNTn Register**

The RxCNTn register is the counter used to indicate the number of received data bytes in the Receiver FIFO which have not been read by the MCU. This register is read only.

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

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **D2~D0**: Receiver FIFO counter

The RxCNTn register is the counter used to indicate the number of Receiver FIFO which is not read by MCU. When Receiver FIFO receives one byte data, the RxCNTn will increase by one; when MCU read one byte data from Receiver FIFO, the RxCNTn will decrease by one. If there are 4 bytes data in the Receiver FIFO, the 5th data will be saved in the shift register. If there is 6th data, the 6th data will saved in the shift register. But the RxCNTn remains the value of 4. The RxCNTn will be clear when reset occurs or UARTEN=1. This register is read only.

Baud Rate Generator

To setup the speed of the serial data communication, the UARTn function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRDHn/BRDLn register and the second is the UARTn modulation control bits, UMODn2~ UMODn0. To prevent accumulated error of the receiver baud rate frequency, it is recommended to use two stop bits for resynchronization after each byte is received. If a baud rate BR is required with UARTn clock f_H.

f_H/BR=Integer Part+Fractional Part

The integer part is loaded into BRDn (BRDHn/BRDLn). The fractional part is multiplied by 8 and rounded, then loaded into UMODn bit field as following:

BRDn=TRUNC(f_H/BR)

UMODn=ROUND [MOD (f_H/BR)×8]

Therefore, the actual baud rate is as following:

Baud rate=f_H/[BRDn+(UMODn/8)]

Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, determine the BRDHn/BRDLn register value, the actual baud rate and the error value for a desired baud rate of 230400.

From the above formula, the BRDn=TRUNC(f_H/BR)=TRUNC(17.36111)=17

The UMODn=ROUND[MOD(f_H/BR)×8]=ROUND(0.36111×8)=ROUND(2.88888)=3

The actual Baud Rate=f_H/[BRDn+(UMODn/8)]=230215.83

Therefore the error is equal to (230215.83-230400)/230400=-0.08%

Modulation Control Example

To get the best-fitting bit sequence for UARTn modulation control bits UMODn2~UMODn0, the following algorithm can be used: Firstly, the fractional part of the theoretical division factor is multiplied by 8. Then the product will be rounded and UMODn2~UMODn0 bits will be filled with the rounded value. The UMODn2~UMODn0 will be added to internal accumulator for every UARTn bit time. Until a carry to bit 3, the corresponding UARTn bit time increases a UARTn clock cycle. The following is an example using the fraction 0.36111 previously calculated: UMODn[2:0]=ROUND(0.36111×8)=011b.

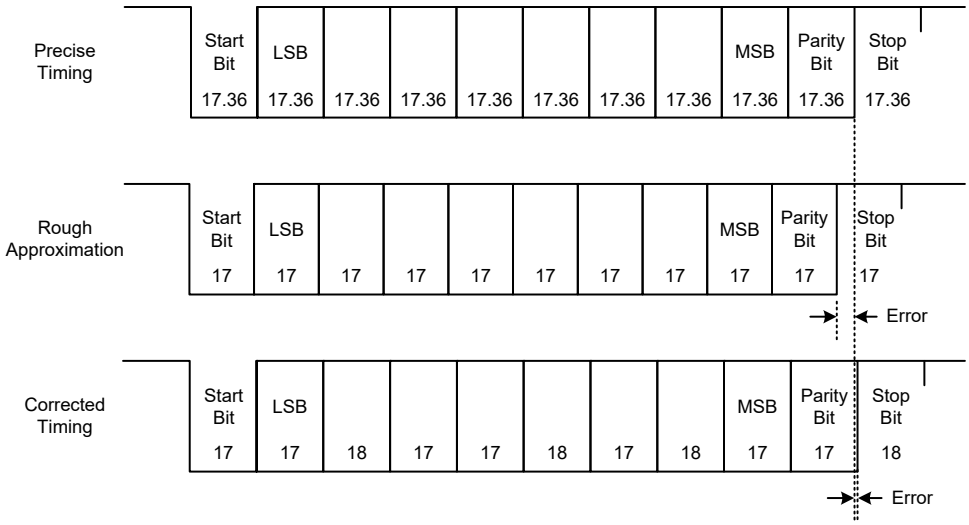
Fraction Addition	Carry to Bit 3	UARTn Bit Time Sequence	Extra UARTn Clock Cycle
0000b+0011b=0011b	No	Start bit	No
0011b+0011b=0110b	No	D0	No
0110b+0011b=1001b	Yes	D1	Yes
1001b+0011b=1100b	No	D2	No
1100b+0011b=1111b	No	D3	No
1111b+0011b=0010b	Yes	D4	Yes
0010b+0011b=0101b	No	D5	No
0101b+0011b=1000b	Yes	D6	Yes
1000b+0011b=1011b	No	D7	No
1011b+0011b=1110b	No	Parity bit	No
1110b+0011b=0001b	Yes	Stop bit	Yes

Baud Rate Correction Example

The following figure presents an example using a baud rate of 230400 generated with UARTn clock f_H. The data format for the following figure is: eight data bits, parity enabled, no address bit, two stop bits.

The following figure shows three different frames:

- The upper frame is the correct one, with a bit-length of 17.36 f_H cycles (4000000/230400=17.36).
- The middle frame uses a rough estimate, with 17 f_H cycles for the bit length.
- The lower frame shows a corrected frame using the best fit for the UARTn modulation control bits UMODn2~UMODn0.



UART Setup and Control

For data transfer, the UARTn function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UARTn hardware, and can be setup to be even, odd, mark, space or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting. The number of data bits along with the parity are setup by programming the BNOn, PRTn1~PRTn0 and PRENn bits. The transmitter always uses two stop bits while the receiver uses one or two stop bits which is determined by the STOPSn bit. The baud rate used to transmit and receive data is setup using the internal 16-bit baud rate generator, while the data is transmitted and received LSB first. Although the UARTn transmitter and receiver are functionally

independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

The basic on/off function of the internal UARTn function is controlled using the UARTENn bit in the UnCR1 register. If the UARTENn, TXENn and RXENn bits are set, then these two UARTn pins will act as normal TXn output pin and RXn/TXn input pin respectively. If no data is being transmitted on the TXn pin, then it will default to a logic high value.

Clearing the UARTENn bit will disable the TXn and RXn/TXn pins. When the UARTn function is disabled, the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UARTn will also reset the error and status flags with bits TXENn, RXENn, TXBRKn, RXIFn, OERRn, FERRn, PERRn and NFn as well as register RxCNT being cleared while bits TIDLEn, TXIFn and RIDLEn will be set. The remaining control bits in the UnCR1, UnCR2, UnCR3, UFCRn, BRDHn and BRDLn registers will remain unaffected. If the UARTENn bit in the UnCR1 register is cleared while the UARTn is active, then all pending transmissions and receptions will be immediately suspended and the UARTn will be reset to a condition as defined above. If the UARTn is then subsequently re-enabled, it will restart again in the same configuration.

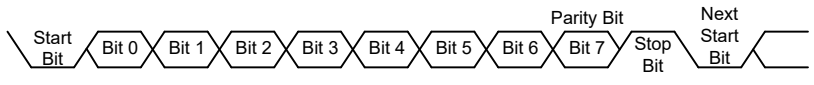
Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UnCR1 and UnCR2 register. The BNOn bit controls the number of data bits which can be set to either 8 or 9, the PRTn1~ PRTn0 bits control the choice of odd, even, mark or space parity, the PRENn bit controls the parity on/off function and the STOPSn bit decides whether one or two stop bits are to be used for the receiver, while the transmitter always uses two stop bits. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only configurable for the receiver. The transmitter uses two stop bits.

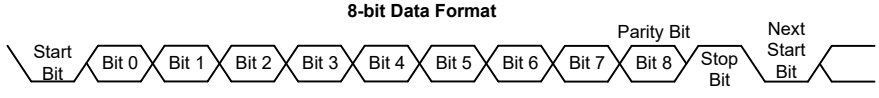
Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1 or 2
1	7	0	1	1 or 2
1	7	1	0	1 or 2
Example of 9-bit Data Formats				
1	9	0	0	1 or 2
1	8	0	1	1 or 2
1	8	1	0	1 or 2

Transmitter Receiver Data Format

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



8-bit Data Format



9-bit Data Format

UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNOn bit in the UnCR1 register. When BNOn 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 TX8n bit in the UnCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSRn, whose data is obtained from the transmit data register, which is known as the TXR_RXRn register. The data to be transmitted is loaded into this TXR_RXRn register by the application program. The TSRn 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 TSRn can then be loaded with new data from the TXR_RXRn register, if it is available. It should be noted that the TSRn 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 TXENn bit is set, but the data will not be transmitted until the TXR_RXRn 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_RXRn register, after which the TXENn bit can be set. When a transmission of data begins, the TSRn is normally empty, in which case a transfer to the TXR_RXRn register will result in an immediate transfer to the TSRn. If during a transmission the TXENn bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared functions by configuring the corresponding pin-shared control bits.

Transmitting Data

When the UARTn is transmitting data, the data is shifted on the TXn pin from the shift register, with the least significant bit first. In the transmit mode, the TXR_RXRn 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 TX8n bit in the UnCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNOn, PRTn1~PRTn0 and PRENn bits to define the required word length and parity type. Two stop bits are used for the transmitter.
- Setup the BRDHn and BRDLn registers and the UMODn2~UMODn0 bits to select the desired baud rate.
- Set the TXENn bit to ensure that the TX pin is used as a UARTn transmitter pin.
- Access the UnSR register and write the data that is to be transmitted into the TXR_RXRn register. Note that this step will clear the TXIFn bit.

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

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

1. An UnSR register access
2. A TXR_RXRn register write execution

The read-only TXIFn flag is set by the UARTn hardware and if set indicates that the TXR_RXRn register is empty and that other data can now be written into the TXR_RXRn register without overwriting the previous data. If the TEIEn bit is set then the TXIFn flag will generate an interrupt.

During a data transmission, a write instruction to the TXR_RXRn register will place the data into the TXR_RXRn register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR_RXRn register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIFn bit being immediately set. When a frame transmission is complete, which happens

after stop bits are sent or after the break frame, the TIDLEn bit will be set. To clear the TIDLEn bit the following software sequence is used:

1. An UnSR register access
2. A TXR_RXRn register write execution

Note that both the TXIFn and TIDLEn bits are cleared by the same software sequence.

Transmitting Break

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

UART Receiver

The UARTn is capable of receiving word lengths of either 8 or 9 bits. If the BNOn bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8n bit of the UnCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSRn. The data which is received on the RXn/TXn external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RXn/TXn pin is sampled for the stop bit, the received data in RSRn is transferred to the receive data register, if the register is empty. The data which is received on the external RXn/TXn input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RXn/TXn pin. It should be noted that the RSRn register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

Receiving Data

When the UARTn receiver is receiving data, the data is serially shifted in on the external RXn/TXn input, LSB first. In the read mode, the TXR_RXRn register forms a buffer between the internal bus and the receiver shift register. The TXR_RXRn register is a four byte deep FIFO data buffer, where four bytes can be held in the FIFO while the fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR_RXRn before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERRn will be subsequently indicated. For continuous multi-byte data transmission, it is strongly recommended that the receiver uses two stop bits to avoid a receiving error caused by the accumulated error of the receiver baud rate frequency.

The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNOn, PRTn1~PRTn0, PRENn and STOPSn bits to define the word length, parity type and number of stop bits.
- Setup the BRDHn, BRDLn and UMODn registers to select the desired baud rate.
- Set the RXENn bit to ensure that the RXn pin is used as a UARTn 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 RXIFn bit in the UnSR register will be set when TXR_RXRn register has data available, the number of the available data bytes can be checked by polling the RxCNT register content.
- When the contents of the shift register have been transferred to the TXR_RXRn register and Receiver FIFO trigger level is reached 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, and then the error flags can be set.

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

1. An UnSR register access
2. A TXR_RXRn register read execution

Receiving Break

Any break character received by the UARTn 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 BNOn bit plus one or two stop bits. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNOn plus one or two stop bits. The RXIFn bit is set, FERRn is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLEn bit is set. A break is regarded as a character that contains only zeros with the FERRn 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 FERRn flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until one or two stop bits are received. It should be noted that the RIDLEn read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UARTn registers will result in the following:

- The framing error flag, FERRn, will be set.
- The receive data register, TXR_RXRn, will be cleared.
- The OERRn, NFn, PERRn, RIDLEn or RXIFn flags will possibly be set.

Idle Status

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

Receiver Interrupt

The read only receive interrupt flag RXIFn in the UnSR register is set by an edge generated by the receiver. An interrupt is generated if RIEn=1, when a word is transferred from the Receive Shift Register, RSRn, to the Receive Data Register, TXR_RXRn. An overrun error can also generate an interrupt if RIEn=1.

When a subroutine will be called with an execution time longer than the time for UARTn to receive five data bytes, if the UARTn received data could not be read in time during the subroutine execution, clear the RXENn bit to zero in advance to suspend data reception. If the UARTn interrupt could not be served in time to process the overrun error during the subroutine execution, ensure that both EMI and RXENn bits are disabled during this period, and then enable EMI and RXENn again after the subroutine execution has been completed to continue the UARTn data reception.

Managing Receiver Errors

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

Overrun Error – OERRn

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

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

- The OERRn flag in the UnSR register will be set.
- The TXR_RXRn contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIEn bit is set.

When the OERRn flag is set to "1", it is necessary to read five data bytes from the four-byte deep receiver FIFO and the shift register immediately to avoid unexpected errors, such as the UARTn is unable to receive data. If such an error occurs, clear the RXENn bit to "0" then set it to "1" again to continue data reception.

The OERRn flag can be cleared by an access to the UnSR register followed by a read to the TXR_RXRn register.

Noise Error – NFn

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

- The read only noise flag, NFn, in the UnSR register will be set on the rising edge of the RXIFn bit.
- Data will be transferred from the Shift register to the TXR_RXRn register.
- No interrupt will be generated. However this bit rises at the same time as the RXIFn bit which itself generates an interrupt.

Note that the NFn flag is reset by a UnSR register read operation followed by a TXR_RXRn register read operation.

Framing Error – FERRn

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

Parity Error – PERRn

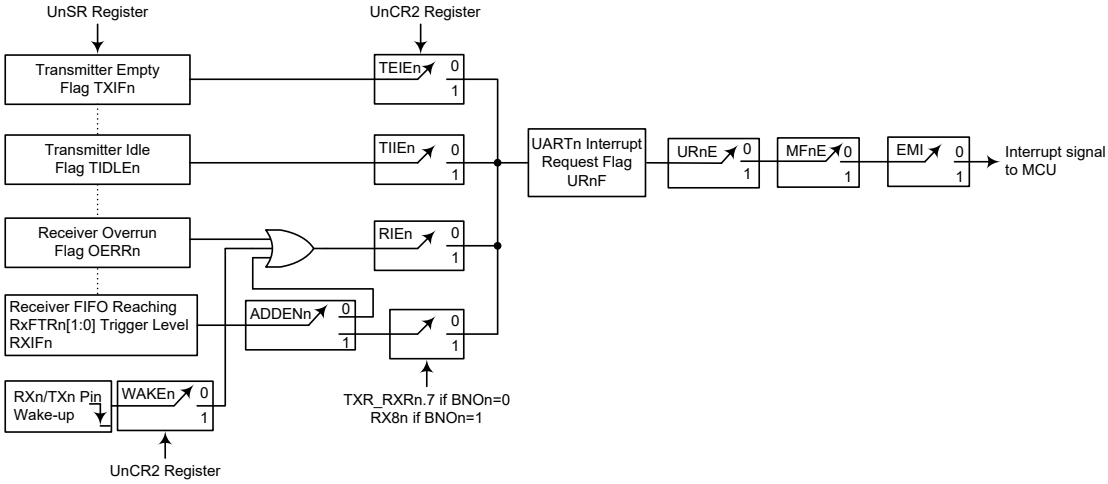
The read only parity error flag, PERRn, in the UnSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PRENn=1, and if the parity type, odd, even, mark or space, is selected. The read only PERRn flag and the received data will be recorded in the UnSR and TXR_RXRn registers respectively. It is cleared on any reset, it should be noted that the flags, FERRn and PERRn, in the UnSR register should first be read by the application program before reading the data word.

UART Interrupt Structure

Several individual UARTn conditions can generate a UARTn interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RXn/TXn pin wake-up. When any of these conditions are created, if the global interrupt enable bit, multi-function interrupt enable bit and its corresponding interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding UnSR register flags which will generate a UARTn interrupt if its associated interrupt enable control bit in the UnCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual UARTn interrupt sources.

The address detect condition, which is also a UARTn interrupt source, does not have an associated flag, but will generate a UARTn interrupt when an address detect condition occurs if its function is enabled by setting the ADDENn bit in the UnCR2 register. An RXn/TXn pin wake-up, which is also a UARTn interrupt source, does not have an associated flag, but will generate a UARTn interrupt if the UARTn clock (f_H) source is switched off and the WAKEn and RIEn bits in the UnCR2 register are set when a falling edge on the RXn/TXn pin occurs.

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



UARTn Interrupt Structure (n=0~1)

Address Detect Mode

Setting the Address Detect Mode bit, ADDENn, in the UnCR2 register, enables this special mode. If this bit is set to 1, then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIFn flag. If the ADDENn bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the URnE, the multi-function interrupt enable bit, MFnE, and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNOn=1 or the 8th bit if BNOn=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 ADDENn bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIFn 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 PRENn to zero.

ADDENn	9th Bit if BNOn=1, 8th Bit if BNOn=0	UARTn Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

ADDENn Bit Function (n=0~1)

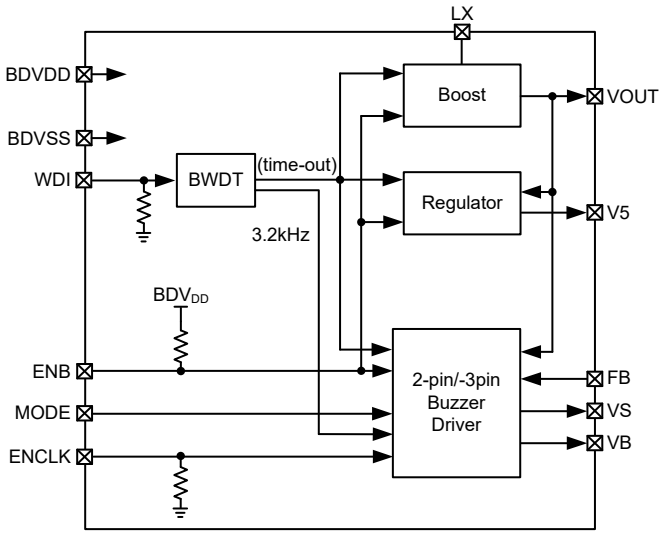
UART Power Down and Wake-up

When the UARTn clock, f_H, is switched off, the UARTn will cease to function. If the MCU switches off the UARTn clock, f_H, and enters the power down mode while a transmission is still in progress, then the transmission will be paused until the UARTn clock source derived from the microcontroller is activated. In a similar way, if the MCU switches off the UARTn clock f_H and enters the IDLE or SLEEP mode by executing the "HALT" instruction while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the UnSR, UnCR1, UnCR2, UnCR3, UFCRn, RxCNTn, TXR_RXRn as well as the BRDHn and BRDLn registers will not be affected. It is recommended to make sure first that the UARTn data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UARTn function contains a receiver RXn/TXn pin wake-up function, which is enabled or disabled by the WAKEn bit in the UnCR2 register. If this bit, along with the UARTn enable bit, UARTENn, the receiver enable bit, RXENn and the receiver interrupt bit, RIEn, are all set when the UART clock (f_H) is off, then a falling edge on the RXn/TXn pin will initiate an RXn/TXn pin wake-up UARTn 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 RXn/TXn pin will be ignored.

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

Buzzer Driver



Buzzer Driver Block Diagram

Functional Description

In order to reduce input voltage drop during startup, when ENB>V_{IH}, the boost convert of buzzer driver has a 250mA peak current limit monitoring for the internal switch, when the soft start current limit threshold is detected, the internal switch is turned off until next cycle, and the boost converter current is unlimited after 25ms.

The buzzer driver provides a Buzzer Watchdog Timer, BWDT, for MCU failure alarm function. It is required to toggle the WDI by inputting a pulse to WDI within the BWDT period, t_{WDI}, to reset the BWDT counter. Otherwise, when the BWDT times out, the internal TO flag will be set high to trigger the buzzer. The TO flag will not cleared to 0 until the MCU toggle the WDI again.

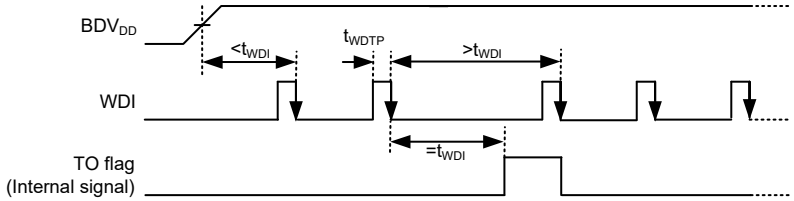
Truth Table

The truth table for the input signal is as follows: When BWDT does not time out, the TO flag is set to 0; when BWDT times out, the TO flag is set to 1. The TO flag has the highest priority.

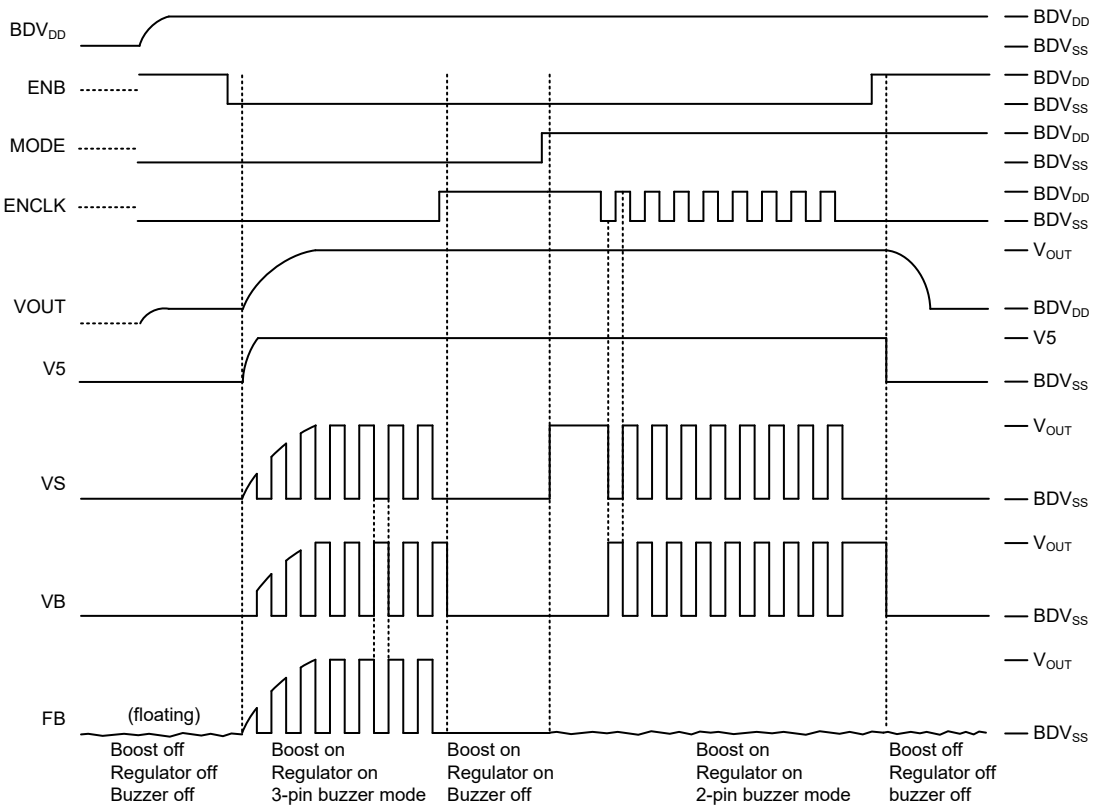
TO flag	ENB	MODE	ENCLK	Function Description
0	0	0	0	Boost on, Regulator on, 3-pin buzzer mode
0	0	0	1	Boost on, Regulator on, Buzzer off
0	0	1	x	Boost on, Regulator on, 2-pin buzzer mode Buzzer signal provided by ENCLK
0	1	x	x	Boost off, Regulator off, Buzzer off
1	x	0	x	Boost on, Regulator on, 3-pin buzzer mode
1	x	1	x	Boost on, Regulator on, 2-pin buzzer mode Buzzer signal provided by BWDT

Timing Chart

WDI Timing



Timing



Low Voltage Detector – LVD

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

LVD Register

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

- **LVDC Register**

Bit	7	6	5	4	3	2	1	0
Name	TLVD1	TLVD0	LVDO	LVDEN	VBGEN	VLVD2	VLVD1	VLVD0
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **TLVD1~TLVD0**: Minimum low voltage width to interrupt time (t_{LVD})
00: (1~2)×t_{LIRC}
01: (3~4)×t_{LIRC}
10: (7~8)×t_{LIRC}
11: (1~2)×t_{LIRC}

Bit 5 **LVDO**: LVD Output Flag
0: No Low Voltage Detected
1: Low Voltage Detected

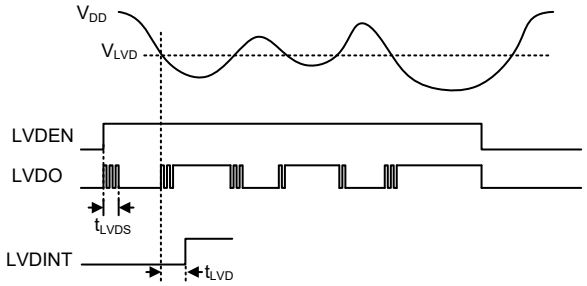
Bit 4 **LVDEN**: Low Voltage Detector Control
0: Disable
1: Enable

Bit 3 **VBGEN**: Bandgap Buffer Control
0: Disable
1: Enable
Note that the Bandgap circuit is enabled when the LVD or the LVR function is enabled or when the VBGEN bit is set high.

Bit 2~0 **VLVD2~VLVD0**: LVD Voltage Selection
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

LVD Operation

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

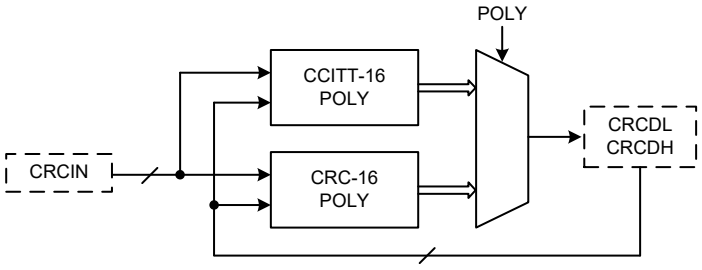


LVD Operation

The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition, i.e., V_{DD} falls below the preset LVD voltage. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated. 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 enter the IDLE Mode.

Cyclic Redundancy Check – CRC

The Cyclic Redundancy Check, CRC, calculation unit is an error detection technique test algorithm and uses to verify data transmission or storage data correctness. A CRC calculation takes a data stream or a block of data as input and generates a 16-bit output remainder. Ordinarily, a data stream is suffixed by a CRC code and used as a checksum when being sent or stored. Therefore, the received or restored data stream is calculated by the same generator polynomial as described in the following section.



CRC Block Diagram

CRC Registers

The CRC generator contains an 8-bit CRC data input register, CRCIN, and a CRC checksum register pair, CRCDH and CRCDL. The CRCIN register is used to input new data and the CRCDH and CRCDL registers are used to hold the previous CRC calculation result. A CRC control register, CRCCR, is used to select which CRC generating polynomial is used.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D7	D6	D5	D4	D3	D2	D1	D0
CRCCR	—	—	—	—	—	—	—	POLY

CRC Register List

• CRCIN Register

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

Bit 7~0 **D7~D0**: CRC input data register

• CRCDL Register

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

Bit 7~0 **D7~D0**: 16-bit CRC checksum low byte data register

• CRCDH Register

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

Bit 7~0 **D15~D8**: 16-bit CRC checksum high byte data register

• CRCCR Register

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **POLY**: 16-bit CRC generating polynomial selection
0: CRC-CCITT: $X^{16}+X^{12}+X^{5}+1$
1: CRC-16: $X^{16}+X^{15}+X^{2}+1$

CRC Operation

The CRC generator provides the 16-bit CRC result calculation based on the CRC16 and CCITT CRC16 polynomials. In this CRC generator, there are only these two polynomials available for the numeric values calculation. It cannot support the 16-bit CRC calculations based on any other polynomials.

The following two expressions can be used for the CRC generating polynomial which is determined using the POLY bit in the CRC control register, CRCCR. The CRC calculation result is called as the CRC checksum, CRCSUM, and stored in the CRC checksum register pair, CRCDH and CRCDL.

- CRC-CCITT: $X^{16}+X^{12}+X^5+1$
- CRC-16: $X^{16}+X^{15}+X^2+1$

CRC Computation

Each write operation to the CRCIN register creates a combination of the previous CRC value stored in the CRCDH and CRCDL registers and the new data input. The CRC unit calculates the CRC data register value is based on byte by byte. It will take one MCU instruction cycle to calculate the CRC checksum.

CRC Calculation Procedures

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Execute an "Exclusive OR" operation with the 8-bit input data byte and the 16-bit CRCSUM high byte. The result is called the temporary CRCSUM.
3. Shift the temporary CRCSUM value left by one bit and move a "0" into the LSB.
4. Check the shifted temporary CRCSUM value after procedure 3.

   If the MSB is 0, then this shifted temporary CRCSUM will be considered as a new temporary CRCSUM.

   Otherwise, execute an "Exclusive OR" operation with the shifted temporary CRCSUM in procedure 3 and a data "8005H". Then the operation result will be regarded as the new temporary CRCSUM.

   Note that the data to be perform an "Exclusive OR" operation is "8005H" for the CRC-16 polynomial while for the CRC-CCITT polynomial the data is "1021H".
5. Repeat the procedure 3 ~ procedure 4 until all bits of the input data byte are completely calculated.
6. Repeat the procedure 2 ~ procedure 5 until all of the input data bytes are completely calculated. Then, the latest calculated result is the final CRC checksum, CRCSUM.

CRC Calculation Examples

- Write 1 byte input data into the CRCIN register and the corresponding CRC checksum are individually calculated as the following table shown.

CRC Data Input / CRC Polynomial	00H	01H	02H	03H	04H	05H	06H	07H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	0000H	1021H	2042H	3063H	4084H	50A5H	60C6H	70E7H
CRC-16 ($X^{16}+X^{15}+X^2+1$)	0000H	8005H	800FH	000AH	801BH	001EH	0014H	8011H

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before each CRC input data is written into the CRCIN register.

- Write 4 bytes input data into the CRCIN register sequentially and the CRC checksum are sequentially listed in the following table.

CRC Data Input / CRC Polynomial	CRCIN=78H→56H→34H→12H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	(CRCDH, CRCDL)=FF9FH→BBC3H→A367H→D0FAH
CRC-16 ($X^{16}+X^{15}+X^2+1$)	(CRCDH, CRCDL)=0110h→91F1h→F2DEh→5C43h

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before the sequential CRC data input operation.

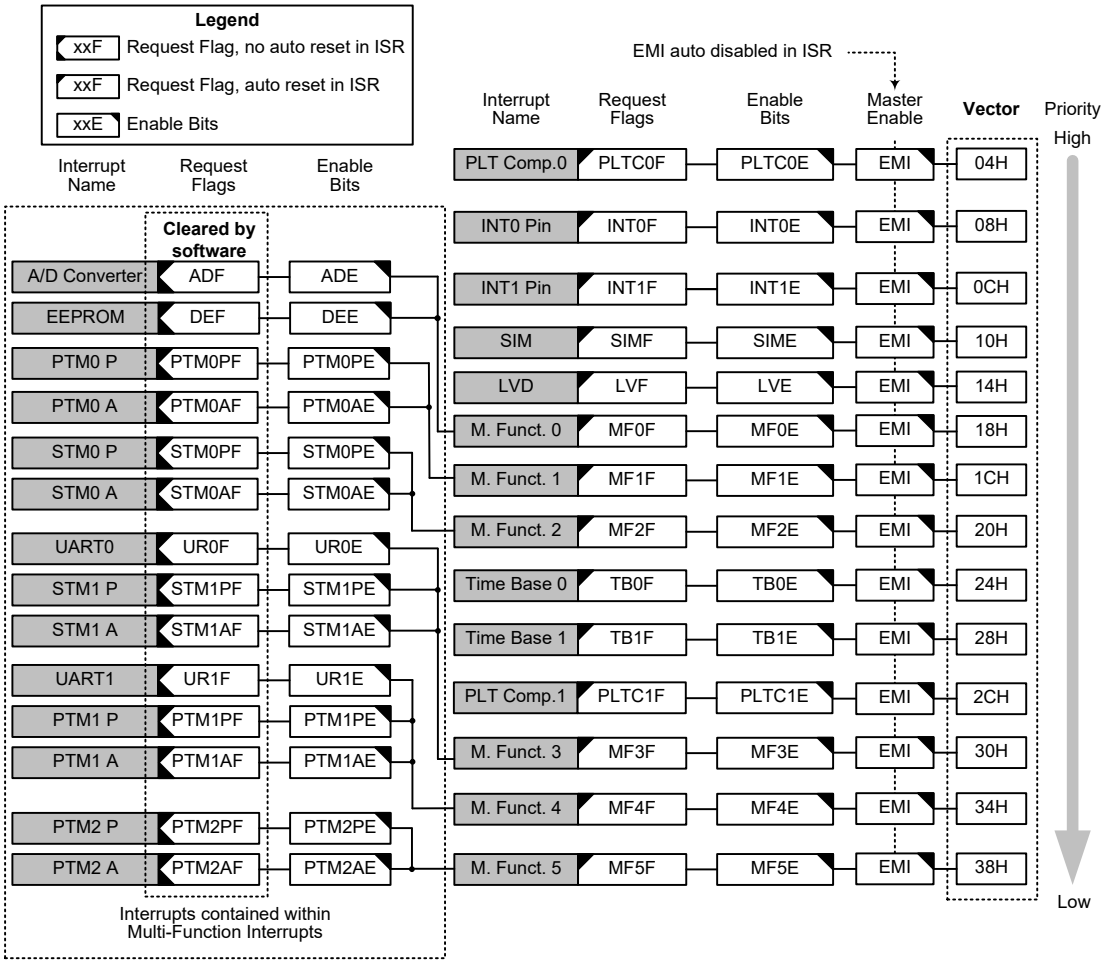
Program Memory CRC Checksum Calculation Example

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Select the CRC-CCITT or CRC-16 polynomial as the generating polynomial using the POLY bit in the CRCCR register.
3. Execute the table read instruction to read the program memory data value.
4. Write the table data low byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.
5. Write the table data high byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.
6. Repeat the procedure 3 ~ procedure 5 to read the next program memory data value and execute the CRC calculation until all program memory data are read followed by the sequential CRC calculation. Then the value in the CRC checksum register pair is the final CRC calculation result.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions including the TMs, Time Bases, SIM, LVD, EEPROM, Power Line Transceiver Comparators and the A/D converter, ect.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector.



Interrupt Structure

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 MFIn register which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt 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
PLT Comparators	PLTCnE	PLTCnF	n=0~1
A/D Converter	ADE	ADF	—
Time Bases	TBnE	TBnF	n=0~1
SIM	SIME	SIMF	—

Function	Enable Bit	Request Flag	Notes
UARTn	URnE	URnF	n=0~1
Multi-function	MFnE	MFnF	n=0~5
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
STMn	STMnPE	STMnPF	n=0~1
	STMnAE	STMnAF	
PTMn	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	—	INT1F	INT0F	PLTC0F	INT1E	INT0E	PLTC0E	EMI
INTC1	MF1F	MF0F	LVF	SIMF	MF1E	MF0E	LVE	SIME
INTC2	PLTC1F	TB1F	TB0F	MF2F	PLTC1E	TB1E	TB0E	MF2E
INTC3	—	MF5F	MF4F	MF3F	—	MF5E	MF4E	MF3E
MFI0	—	—	DEF	ADF	—	—	DEE	ADE
MFI1	—	—	PTM0AF	PTM0PF	—	—	PTM0AE	PTM0PE
MFI2	—	—	STM0AF	STM0PF	—	—	STM0AE	STM0PE
MFI3	—	UR0F	STM1AF	STM1PF	—	UR0E	STM1AE	STM1PE
MFI4	—	UR1F	PTM1AF	PTM1PF	—	UR1E	PTM1AE	PTM1PE
MFI5	—	—	PTM2AF	PTM2PF	—	—	PTM2AE	PTM2PE

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 trigger edge selection for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt trigger edge selection 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	—	INT1F	INT0F	PLTC0F	INT1E	INT0E	PLTC0E	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 **INT1F**: INT1 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **INT0F**: INT0 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PLTC0F**: PLT Comparator 0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **INT1E**: INT1 interrupt control
0: Disable
1: Enable

Bit 2 **INT0E**: INT0 interrupt control
0: Disable
1: Enable

Bit 1 **PLTC0E**: PLT Comparator 0 interrupt control
0: Disable
1: Enable

Bit 0 **EMI**: Global interrupt control
0: Disable
1: Enable

• **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	MF1F	MF0F	LVF	SIMF	MF1E	MF0E	LVE	SIME
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 **MF1F**: Multi-function 1 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **MF0F**: Multi-function 0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **LVF**: LVD interrupt request flag
0: No request
1: Interrupt request

Bit 4 **SIMF**: SIM interrupt request flag
0: No request
1: Interrupt request

Bit 3 **MF1E**: Multi-function 1 interrupt control
0: Disable
1: Enable

Bit 2 **MF0E**: Multi-function 0 interrupt control
0: Disable
1: Enable

Bit 1 **LVE**: LVD interrupt control
0: Disable
1: Enable

Bit 0 **SIME**: SIM interrupt control
0: Disable
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	PLTC1F	TB1F	TB0F	MF2F	PLTC1E	TB1E	TB0E	MF2E
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 **PLTC1F**: PLT Comparator 1 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **MF2F**: Multi-function 2 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **PLTC1E**: PLT Comparator 1 interrupt control
0: Disable
1: Enable

Bit 2 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

Bit 1 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

Bit 0 **MF2E**: Multi-function 2 interrupt control
0: Disable
1: Enable

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	MF5F	MF4F	MF3F	—	MF5E	MF4E	MF3E
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 **MF5F**: Multi-function 5 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **MF4F**: Multi-function 4 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **MF3F**: Multi-function 3 interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **MF5E**: Multi-function 5 interrupt control
0: Disable
1: Enable

Bit 1 **MF4E**: Multi-function 4 interrupt control
0: Disable
1: Enable

Bit 0 **MF3E**: Multi-function 3 interrupt control
0: Disable
1: Enable

• **MFI0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	DEF	ADF	—	—	DEE	ADE
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
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **ADF**: A/D converter interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **DEE**: Data EEPROM interrupt control
0: Disable
1: Enable

Bit 0 **ADE**: A/D converter interrupt control
0: Disable
1: Enable

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM0AF	PTM0PF	—	—	PTM0AE	PTM0PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PTM0AF**: PTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **PTM0PF**: PTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTM0AE**: PTM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTM0PE**: PTM0 Comparator P match interrupt control
0: Disable
1: Enable

• **MFI2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	STM0AF	STM0PF	—	—	STM0AE	STM0PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **STM0AF**: STM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **STM0PF**: STM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **STM0AE**: STM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **STM0PE**: STM0 Comparator P match interrupt control
0: Disable
1: Enable

• **MFI3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	UR0F	STM1AF	STM1PF	—	UR0E	STM1AE	STM1PE
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 **UR0F**: UART0 interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 5 **STM1AF**: STM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **STM1PF**: STM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3 Unimplemented, read as "0"

Bit 2 **UR0E**: UART0 interrupt control
0: Disable
1: Enable

Bit 1 **STM1AE**: STM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **STM1PE**: STM1 Comparator P match interrupt control
0: Disable
1: Enable

• **MFI4 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	UR1F	PTM1AF	PTM1PF	—	UR1E	PTM1AE	PTM1PE
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 **UR1F**: UART1 interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 5 **PTM1AF**: PTM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **PTM1PF**: PTM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3 Unimplemented, read as "0"

Bit 2 **UR1E**: UART1 interrupt control
0: Disable
1: Enable

Bit 1 **PTM1AE**: PTM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTM1PE**: PTM1 Comparator P match interrupt control
0: Disable
1: Enable

• **MFI5 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM2AF	PTM2PF	—	—	PTM2AE	PTM2PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PTM2AF**: PTM2 Comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **PTM2PF**: PTM2 Comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTM2AE**: PTM2 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTM2PE**: PTM2 Comparator P match interrupt control
0: Disable
1: Enable

Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P, Comparator A match or A/D conversion completion etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high, then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a "JMP" which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a "RETI", which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

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

External Interrupts

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 their respective interrupt vector addresses, 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 corresponding 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.

PLT Comparator Interrupts

The PLT comparator interrupts are controlled by the Power Line Transceiver circuit internal comparators. A PLT comparator interrupt request will take place when the PLT comparator interrupt request flag, PLTCnF, is set, a situation that will occur when the PLT comparator output bit changes state. To allow the program to branch to their interrupt vector addresses, the global interrupt enable bit, EMI, and PLT comparator interrupt enable bit, PLTCnE, must first be set. When the interrupt is enabled, the stack is not full and the PLT comparator inputs generate a comparator output bit transition, a subroutine call to the PLT comparator interrupt vector, will take place. When the interrupt is serviced, the PLT comparator interrupt request flag, PLTCnF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

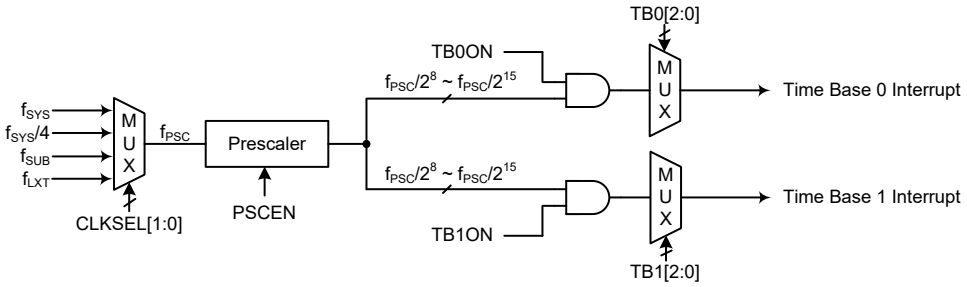
A/D Converter Interrupt

The A/D Converter Interrupt is contained within the Multi-function Interrupt. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the Multi-function Interrupt vector, will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the ADF flag will not be automatically cleared, it has to be cleared by the application program.

Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happen, their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, 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}, $f_{SYS}/4$, f_{SUB} or f_{LXT} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C or TB1C register to obtain longer interrupt periods whose value ranges. The clock cource that generate f_{PSC}, which in turn controls the Time Base interrupt period, is selected using the CLKSEL[1:0] bits in the PSCR register.



Time Base Interrupt

• **PSCR Register**

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

Bit 7~3 Unimplemented, read as "0"

Bit 2 **PSCEN**: Prescaler clock enable control
0: Disable
1: Enable
The PSCEN bit is the Prescaler clock enable or disable control bit. When the Prescale clock is disabled, it can reduce extra power consumption.

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source f_{PSC} selection
00: f_{SYS}
01: $f_{SYS}/4$
10: f_{SUB}
11: f_{LXT}

• **TBnC Register (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	TBnON	—	—	—	—	TBn2	TBn1	TBn0
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TBnON**: Time Base n Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TBn2~TBn0**: Select Time Base n 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: $2^{15}/f_{PSC}$

Multi-function Interrupts

Within the device there are several Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the ADC interrupt, Data EEPROM interrupt, PTM interrupts, STM interrupts and UART interrupts.

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

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

Serial Interface Module Interrupt

The Serial Interface Module Interrupt is also known as the SIM Interrupt. A SIM Interrupt repuest 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 an I^2C time-out situation has occurred. To allow the program to branch to its 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 interrupt is serviced, the Serial Interface Interrupt flag, SIMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

UART Interrupts

The UART Interrupts are contained within the Multi-function Interrupts. 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 reaching FIFO trigger level, receiver overrun, address detect and an RXn/TXn pin wake-up. To allow the program to branch to the respective interrupt vector addresses, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URnE, and associated Multi-function interrupt enable bit, 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 Multi-function Interrupt vector will take place. When the Interrupt is serviced, the EMI bit will also be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the URnF flag will not be automatically cleared, it has to be cleared by the application program.

However, the UnSR register flags will only be cleared when certain actions are taken by the UARTn, the details of which are given in the UART section.

LVD Interrupt

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

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 interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, and its associated multi-function interrupt request flag 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 relevant Multi-function Interrupt vector locations, will take place.When the EEPROM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the Multi-function interrupt request flag will be automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

TM Interrupts

The Standard and Periodic Type TMs each has two interrupts. All of the TM interrupts are contained within the Multi-function Interrupts. For all of the TM types there are two interrupt request flags and two enable bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector 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 flag 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 enter the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program.

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.

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. All options must be defined for proper system function, the details of which are shown in the table.

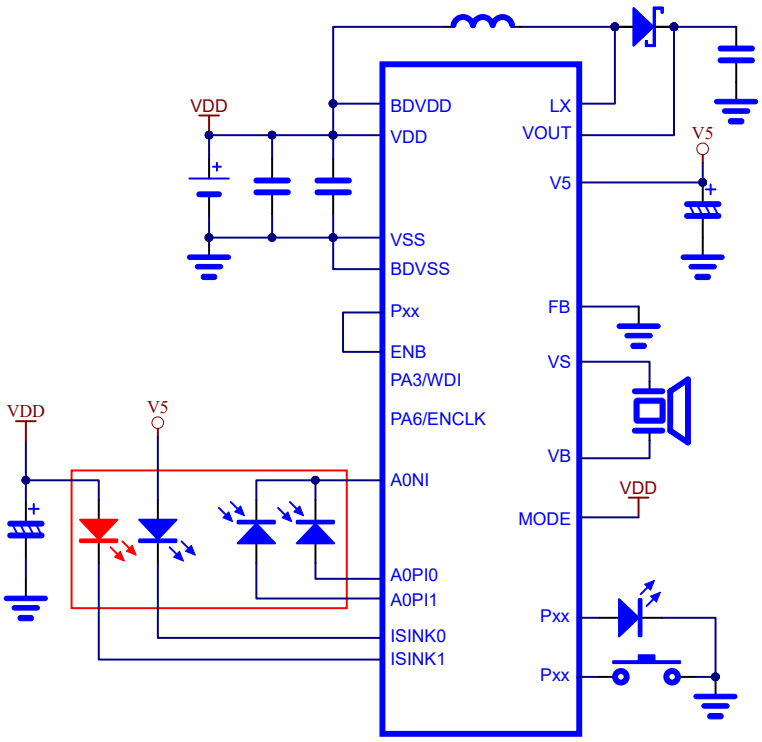
No.	Options
Oscillator Option	
1	HIRC frequency selection – f_{HIRC}: 2MHz, 4MHz or 8MHz
Temperature Sensor Option	
2	Temperature Calibration Selection 1: Disable 2: Enable

Note: 1. When the HIRC has been configured at a frequency shown in this table, the HIRC1 and HIRC0 bits should also be set to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

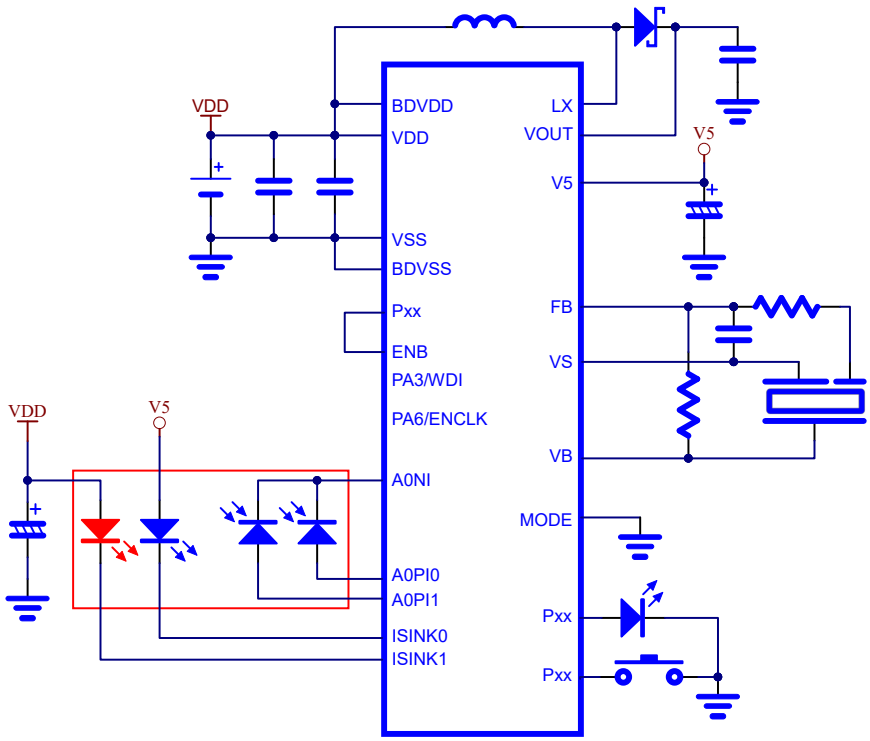
2. If it is required to implement temperature calibration and "Enable" has been selected, when using the writer for programming, it will detect whether the temperature module (EMDE001A) is connected or not. If the temperature module is not connected, the writer will indicate an error and cannot implement programming.

Application Circuits

2-pin Buzzer Mode



3-pin Buzzer Mode



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 several kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
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 (specific page) 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 two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This cannot only reduce the use of Flash memory space but also improve the CPU execution efficiency.

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 (specific page) 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 (specific page) 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 three cycles are required, if no skip takes place two cycles is required.

2. Any extended instruction which changes the contents of the PCL register will also require three cycles for execution.

Instruction Definition

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] ← [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 ← [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 ← ACC − [m] − 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 ← ACC – [m] – 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] ← ACC − [m] − 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 bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m]≠ 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 The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

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 (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 (specific page) to TBLH and Data Memory Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

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] ← [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 ← [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] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

LDEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation [m] ← [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 ← [m] − 1

Affected flag(s) Z

LINC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation [m] ← [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 ← [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 theAccumulator 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 ← ACC − [m] − 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] ← ACC − [m] − 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 three 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 three 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 three 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 three 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 bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the content of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m] ≠ 0

Affected flag(s) None

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 The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

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 three 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 three cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

LTABRD [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 (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

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 (specific page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)

TBLH ← program code (high byte)

Affected flag(s) None

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

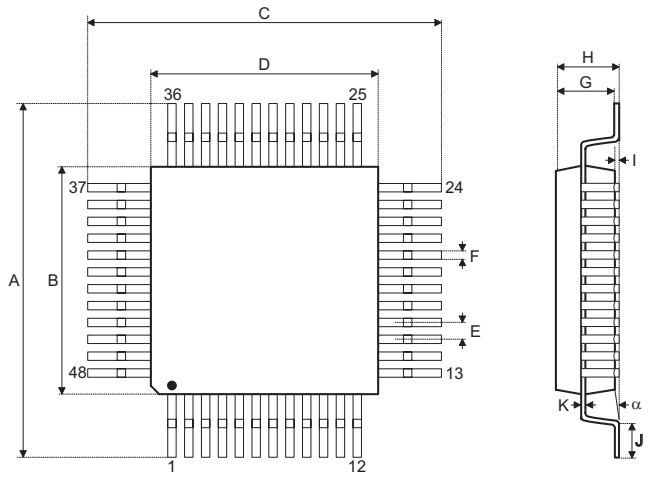
Package Information

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.

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- 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	
D2	0.170	—	0.211
E2	0.170	—	0.211
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	
D2	4.31	—	5.36
E2	4.31	—	5.36
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°