Touch A/D Flash MCU with LCD Driver

BS67F360CA

Revision: V1.20 Date: February 05, 2026

www.holtek.com

Features

CPU Features

- Holtek TinyPower™ Technology
- Operating voltage
  - f_{SYS}=8MHz: 1.8V~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 8/12/16MHz RC – HIRC
  - External Low Speed 32.768kHz Crystal – LXT
  - Internal Low Speed 32kHz RC – LIRC
- 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
- 16-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 16K×16
- Data Memory: 1024×8
- True EEPROM Memory: 1024×8
- In Application Programming function – IAP
- 128-bit unique ID
- Watchdog Timer function
- Up to 59 bidirectional I/O lines
- Programmable I/O port source current for LED driving applications
- 2 external interrupt lines shared with I/O pins
- 7 Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
  - Three 10-bit Compact Type TMs
  - One 16-bit Standard Type TM
  - Three 10-bit Periodic Type TMs (PTM1 supports complementary PWM output with dead time control)
- Serial Interface Module – SIM includes SPI and I^2C interfaces
- One independent I^2C interface
- Three Fully-duplex/Half-duplex Universal Asynchronous Receiver and Transmitter Interfaces – UART0~UART2

- Dual Time Base functions for generation of fixed time interrupt signals
- 24 touch key functions – fully integrated without requiring external components
- Over Voltage Protection function – OVP
- 8 external channel 12-bit resolution A/D converter with temperature sensor and internal reference voltage V_{VR}
- LCD Driver function
  - SEGs × COMs: 43×4, 41×6 or 39×8
  - Duty type: 1/4 duty, 1/6 duty or 1/8 duty
  - Bias level: 1/3 bias or 1/4 bias
  - Bias type: R type or C type
  - Waveform type: type A or type B
- Integrated 16-bit Cyclic Redundancy Check function – CRC
- Low voltage reset function
- Low voltage detect function
- Package types: 48/64-pin LQFP

General Description

The device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller with Touch Key module and LCD driver, which is specifically designed for Touch and LCD display applications.

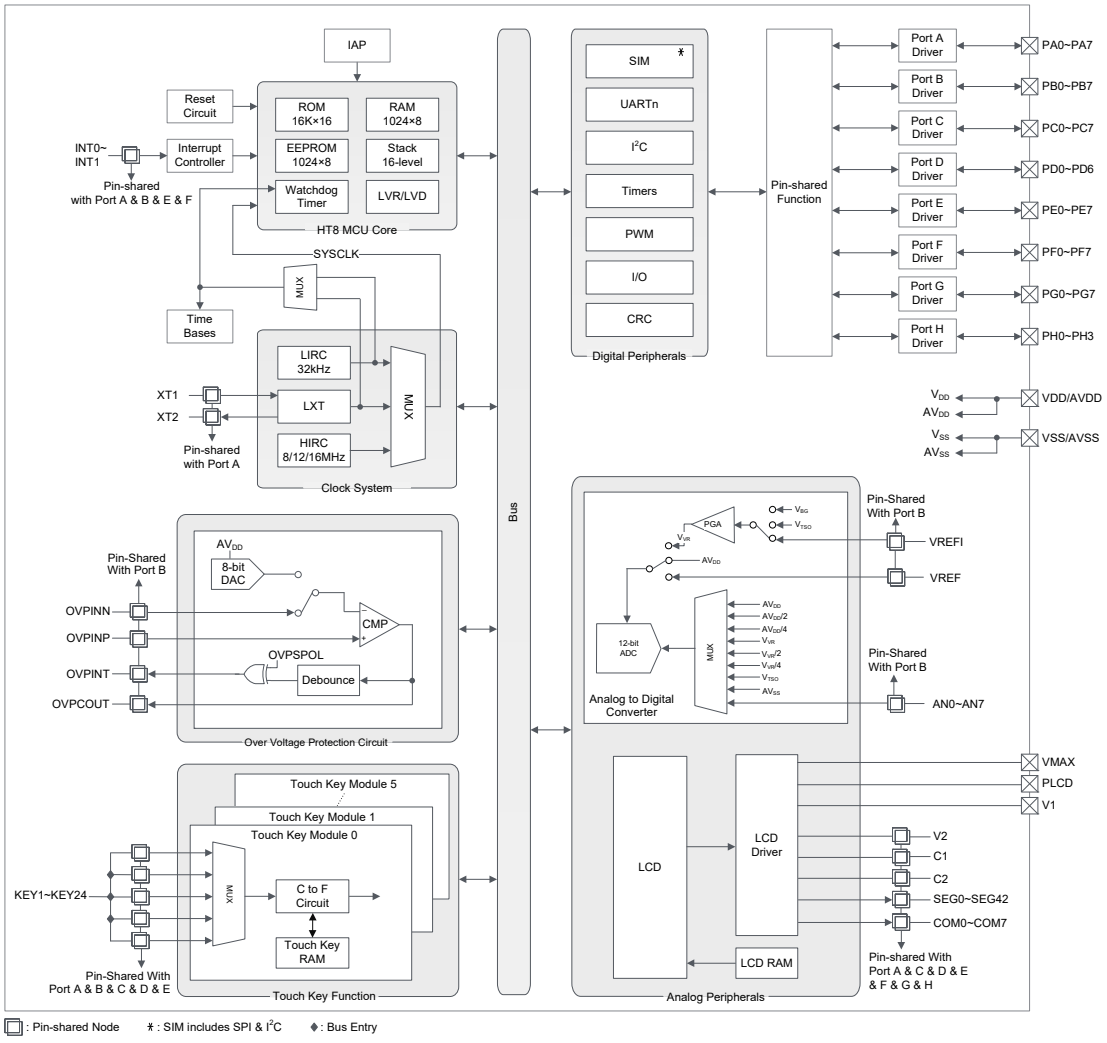
For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc. By using the In Application Programming 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 12-bit A/D converter, an over voltage protection function, fully integrated touch key function and LCD driver. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions etc. Communication with the outside world is catered for by including fully integrated SPI, I^2C and UART interface functions, these popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detect coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of external low, internal high and low 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.

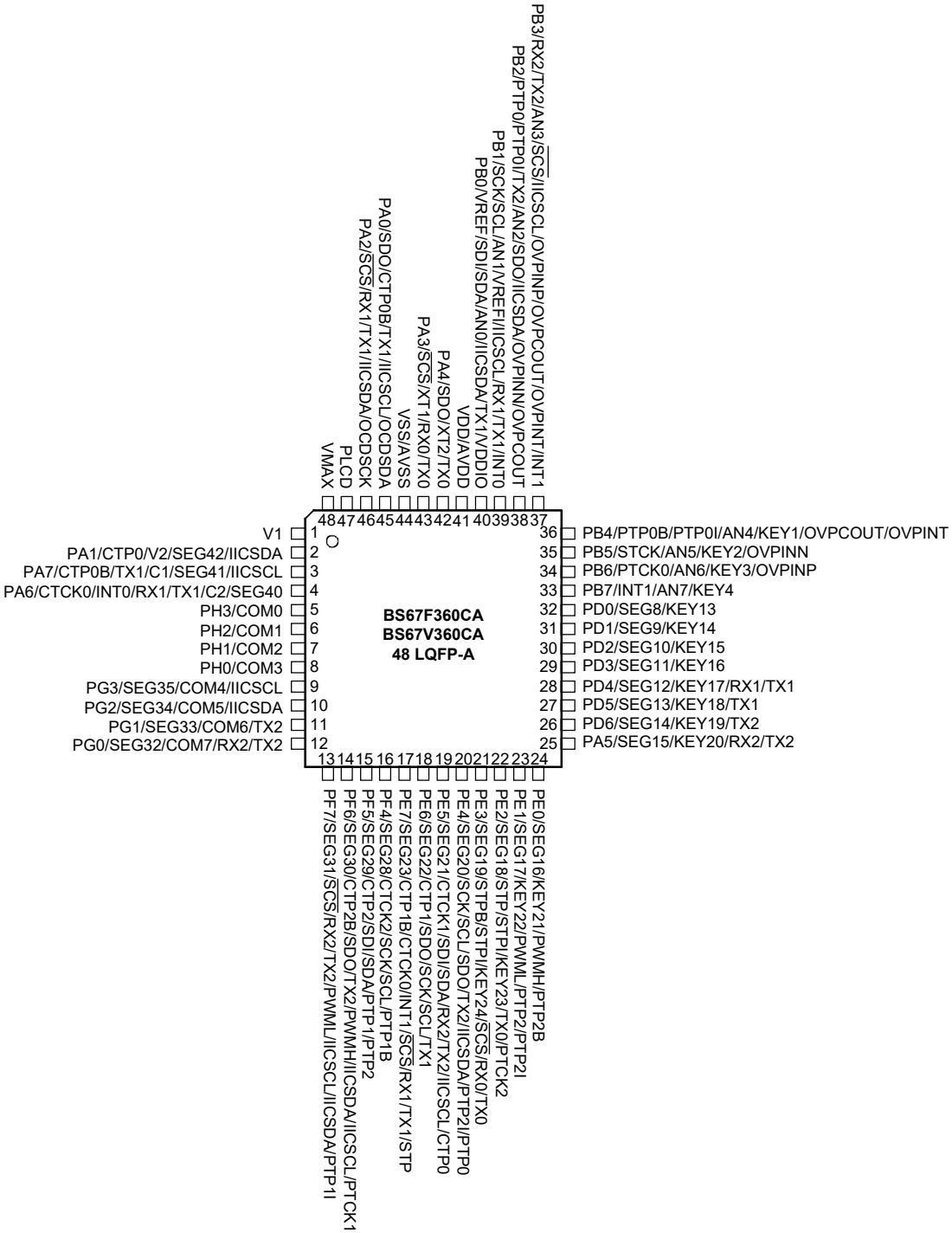
The inclusion of flexible I/O programming features, Time Base functions, Cyclic Redundancy Check function along with many other features ensure that the device will find excellent use in a huge range of modern Touch Key product applications such as thermostats, and household appliances such as electric rice cookers, dehumidifiers, air purifiers, and so on.

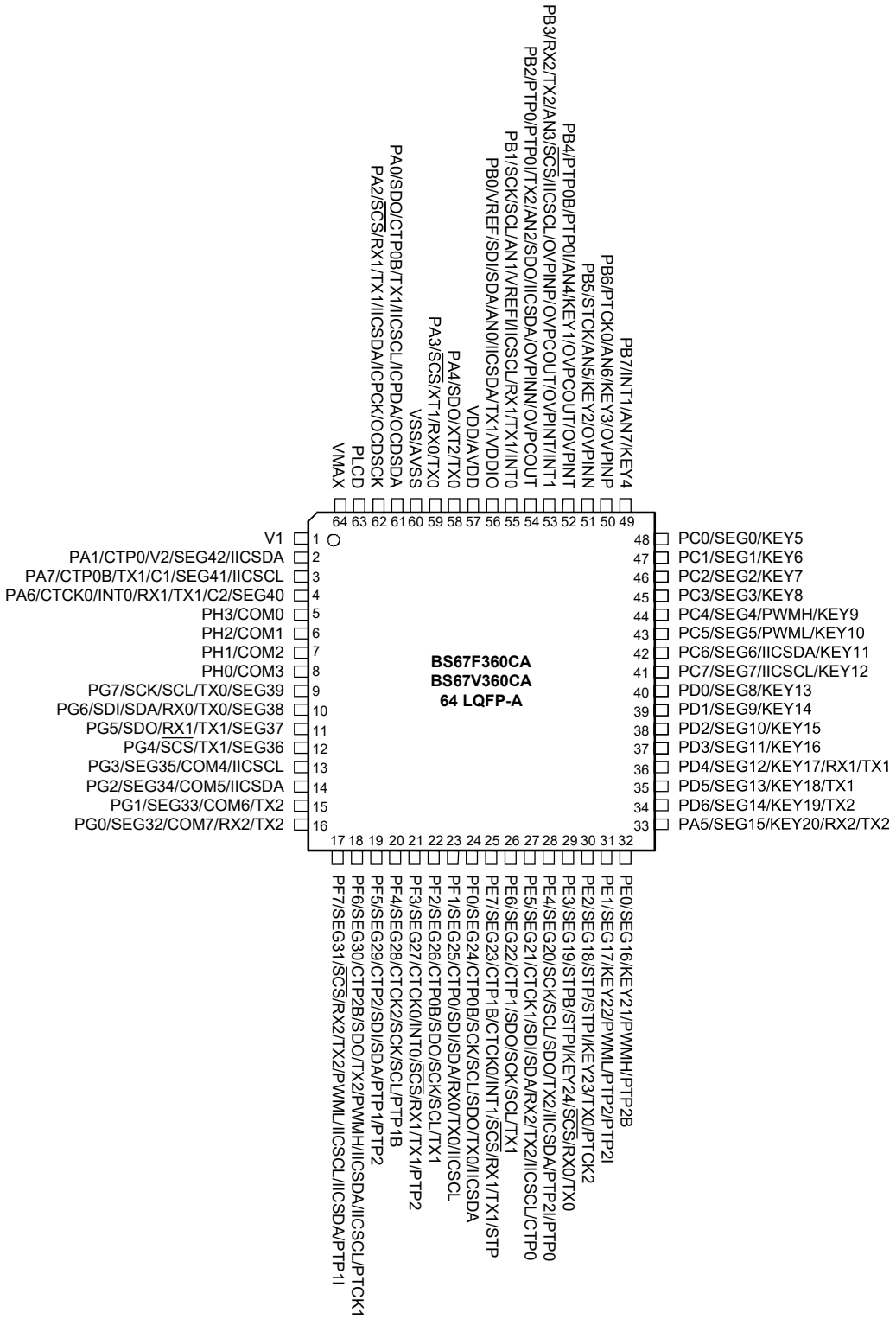
Block Diagram



: Pin-shared Node ✱ : SIM includes SPI & I²C ◆ : Bus Entry

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 for the OCDS dedicated pins and as such only available for the BS67V360CA device which is the OCDS EV chip for the BS67F360CA 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/SDO/CTP0B/TX1/IICSCL/ ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SDO	PAS0	—	CMOS	SIM SPI serial data output
	CTP0B	PAS0	—	CMOS	CTM0 inverted output
	TX1	PAS0	—	CMOS	UART1 serial data output
	IICSCL	PAS0 IFS3	ST	NMOS	Independent I^2C clock line
	ICPDA	—	ST	CMOS	ICP data/address pin
	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
PA1/CTP0/V2/SEG42/IICSDA	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP0	PAS0	—	CMOS	CTM0 output
	V2	PAS0	PWR	AN	LCD voltage pump
	SEG42	PAS0	—	AN	LCD Segment output
	IICSDA	PAS0 IFS3	ST	NMOS	Independent I^2C data line
PA2/$\overline{SCS}$/RX1/TX1/IICSDA/ ICPCK/OCDSCK	PA2	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	$\overline{SCS}$	PAS1 IFS4	ST	CMOS	SIM SPI slave select pin
	RX1/TX1	PAS1 IFS1	ST	CMOS	UART1 serial data input in full-duplex communication or UART1 serial data input / output in Single Wire Mode communication
	IICSDA	PAS1 IFS3	ST	NMOS	Independent I^2C data line
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/$\overline{SCS}$/XT1/RX0/TX0	PA3	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	$\overline{SCS}$	PAS1 IFS4	ST	CMOS	SIM SPI slave select pin
	XT1	PAS1	AN	—	LXT oscillator pin
	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

Pin Name	Function	OPT	I/T	O/T	Description
PA4/SDO/XT2/TX0	PA4	PAPU PAWU PAS2	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SDO	PAS2	—	CMOS	SIM SPI serial data output
	XT2	PAS2	—	AN	LXT oscillator pin
	TX0	PAS2	—	CMOS	UART0 serial data output
PA5/SEG15/KEY20/RX2/TX2	PA5	PAPU PAWU PAS2	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SEG15	PAS2	—	AN	LCD Segment output
	KEY20	PAS2	AN	—	Touch key input
	RX2/TX2	PAS2 IFS2	ST	CMOS	UART2 serial data input in full-duplex communication or UART2 serial data input / output in Single Wire Mode communication
PA6/CTCK0/INT0/RX1/TX1/C2/ SEG40	PA6	PAPU PAWU PAS3	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK0	PAS3 IFS2	ST	—	CTM0 clock input
	INT0	PAS3 INTEG INTC0 IFS1	ST	—	External interrupt input
	RX1/TX1	PAS3 IFS1	ST	CMOS	UART1 serial data input in full-duplex communication or UART1 serial data input / output in Single Wire Mode communication
	C2	PAS3	AN	AN	LCD voltage pump
	SEG40	PAS3	—	AN	LCD Segment output
PA7/CTP0B/TX1/C1/SEG41/ IICSCL	PA7	PAPU PAWU PAS3	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP0B	PAS3	—	CMOS	CTM0 inverted output
	TX1	PAS3	—	CMOS	UART1 serial data output
	C1	PAS3	AN	AN	LCD voltage pump
	SEG41	PAS3	—	AN	LCD Segment output
	IICSCL	PAS3 IFS3	ST	NMOS	Independent I^2C clock line
PB0/VREF/SDI/SDA/ AN0/IICSDA/TX1/VDDIO	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	VREF	PBS0	AN	—	A/D converter external reference input
	SDI	PBS0 IFS0	ST	—	SIM SPI serial data input
	SDA	PBS0 IFS0	ST	NMOS	SIM I^2C data line
	AN0	PBS0	AN	—	A/D converter external channel input
	IICSDA	PBS0 IFS3	ST	NMOS	Independent I^2C data line
	TX1	PBS0	—	CMOS	UART1 serial data output
	VDDIO	PBS0	PWR	—	Power supply for PB1~PB3

Pin Name	Function	OPT	I/T	O/T	Description
PB1/SCK/SCL/AN1/VREFI/ IICSCL/RX1/TX1/INT0	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SCK	PBS0 IFS0	ST	CMOS	SIM SPI serial clock
	SCL	PBS0 IFS0	ST	NMOS	SIM I²C clock line
	AN1	PBS0	AN	—	A/D converter external channel input
	VREFI	PBS0	AN	—	A/D converter PGA input
	IICSCL	PBS0 IFS3	ST	NMOS	Independent I²C clock line
	RX1/TX1	PBS0 IFS1	ST	CMOS	UART1 serial data input in full-duplex communication or UART1 serial data input / output in Single Wire Mode communication
	INT0	PBS0 INTEG INTC0 IFS1	ST	—	External interrupt input
PB2/PTP0/PTP0I/TX2/ AN2/SDO/IICSDA/ OVPINN/OVPCOUT	PB2	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP0	PBS1	—	CMOS	PTM0 output
	PTP0I	PBS1 IFS2	ST	—	PTM0 capture input
	TX2	PBS1	—	CMOS	UART2 serial data output
	AN2	PBS1	AN	—	A/D converter external channel input
	SDO	PBS1	—	CMOS	SIM SPI serial data output
	IICSDA	PBS1 IFS3	ST	NMOS	Independent I²C data line
	OVPINN	PBS1	AN	—	OVP comparator negative input
	OVPCOUT	PBS1	—	CMOS	OVP comparator output (before debounce)
PB3/RX2/TX2/AN3/ $\overline{SCS}$/IICSCL/OVPINP/ OVPCOUT/OVPINT/INT1	PB3	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	RX2/TX2	PBS1 IFS2	ST	CMOS	UART2 serial data input in full-duplex communication or UART2 serial data input / output in Single Wire Mode communication
	AN3	PBS1	AN	—	A/D converter external channel input
	$\overline{SCS}$	PBS1 IFS4	ST	CMOS	SIM SPI slave select pin
	IICSCL	PBS1 IFS3	ST	NMOS	Independent I²C clock line
	OVPINP	PBS1	AN	—	OVP comparator positive input
	OVPCOUT	PBS1	—	CMOS	OVP comparator output (before debounce)
	OVPINT	PBS1	—	CMOS	OVP comparator output (after debounce)
	INT1	PBS1 INTEG INTC0 IFS4	ST	—	External interrupt input

Pin Name	Function	OPT	I/T	O/T	Description
PB4/PTP0B/PTP0I/AN4/KEY1/OVPCOUT/OVPINT	PB4	PBPU PBS2	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP0B	PBS2	—	CMOS	PTM0 inverted output
	PTP0I	PBS2 IFS2	ST	—	PTM0 capture input
	AN4	PBS2	AN	—	A/D converter external channel input
	KEY1	PBS2	AN	—	Touch key input
	OVPCOUT	PBS2	—	CMOS	OVP comparator output (before debounce)
	OVPINT	PBS2	—	CMOS	OVP comparator output (after debounce)
PB5/STCK/AN5/KEY2/OVPINN	PB5	PBPU PBS2	ST	CMOS	General purpose I/O. Register enabled pull-high
	STCK	PBS2	ST	—	STM clock input
	AN5	PBS2	AN	—	A/D converter external channel input
	KEY2	PBS2	AN	—	Touch key input
	OVPINN	PBS2	AN	—	OVP comparator negative input
PB6/PTCK0/AN6/KEY3/OVPINP	PB6	PBPU PBS3	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTCK0	PBS3	ST	—	PTM0 clock/capture input
	AN6	PBS3	AN	—	A/D converter external channel input
	KEY3	PBS3	AN	—	Touch key input
	OVPINP	PBS3	AN	—	OVP comparator positive input
PB7/INT1/AN7/KEY4	PB7	PBPU PBS3	ST	CMOS	General purpose I/O. Register enabled pull-high
	INT1	PBS3 INTEG INTC0 IFS4	ST	—	External interrupt input
	AN7	PBS3	AN	—	A/D converter external channel input
	KEY4	PBS3	AN	—	Touch key input
PC0/SEG0/KEY5	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG0	PCS0	—	AN	LCD Segment output
	KEY5	PCS0	AN	—	Touch key input
PC1/SEG1/KEY6	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG1	PCS0	—	AN	LCD Segment output
	KEY6	PCS0	AN	—	Touch key input
PC2/SEG2/KEY7	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG2	PCS0	—	AN	LCD Segment output
	KEY7	PCS0	AN	—	Touch key input
PC3/SEG3/KEY8	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG3	PCS0	—	AN	LCD Segment output
	KEY8	PCS0	AN	—	Touch key input
PC4/SEG4/PWMH/KEY9	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG4	PCS1	—	AN	LCD Segment output
	PWMH	PCS1	—	CMOS	PWM output
	KEY9	PCS1	AN	—	Touch key input

Pin Name	Function	OPT	I/T	O/T	Description
PC5/SEG5/PWML/KEY10	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG5	PCS1	—	AN	LCD Segment output
	PWML	PCS1	—	CMOS	PWM complementary output
	KEY10	PCS1	AN	—	Touch key input
PC6/SEG6/IICSDA/KEY11	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG6	PCS1	—	AN	LCD Segment output
	IICSDA	PCS1 IFS3	ST	NMOS	Independent I^2C data line
	KEY11	PCS1	AN	—	Touch key input
PC7/SEG7/IICSCL/KEY12	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG7	PCS1	—	AN	LCD Segment output
	IICSCL	PCS1 IFS3	ST	NMOS	Independent I^2C clock line
	KEY12	PCS1	AN	—	Touch key input
PD0/SEG8/KEY13	PD0	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG8	PDS0	—	AN	LCD Segment output
	KEY13	PDS0	AN	—	Touch key input
PD1/SEG9/KEY14	PD1	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG9	PDS0	—	AN	LCD Segment output
	KEY14	PDS0	AN	—	Touch key input
PD2/SEG10/KEY15	PD2	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG10	PDS0	—	AN	LCD Segment output
	KEY15	PDS0	AN	—	Touch key input
PD3/SEG11/KEY16	PD3	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG11	PDS0	—	AN	LCD Segment output
	KEY16	PDS0	AN	—	Touch key input
PD4/SEG12/KEY17/RX1/TX1	PD4	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG12	PDS1	—	AN	LCD Segment output
	KEY17	PDS1	AN	—	Touch key input
	RX1/TX1	PDS1 IFS1	ST	CMOS	UART1 serial data input in full-duplex communication or UART1 serial data input / output in Single Wire Mode communication
PD5/SEG13/KEY18/TX1	PD5	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG13	PDS1	—	AN	LCD Segment output
	KEY18	PDS1	AN	—	Touch key input
	TX1	PDS1	—	CMOS	UART1 serial data output
PD6/SEG14/KEY19/TX2	PD6	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG14	PDS1	—	AN	LCD Segment output
	KEY19	PDS1	AN	—	Touch key input
	TX2	PDS1	—	CMOS	UART2 serial data output

Pin Name	Function	OPT	I/T	O/T	Description
PE0/SEG16/KEY21/ PWMH/PTP2B	PE0	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG16	PES0	—	AN	LCD Segment output
	KEY21	PES0	AN	—	Touch key input
	PWMH	PES0	—	CMOS	PWM output
	PTP2B	PES0	—	CMOS	PTM2 inverted output
PE1/SEG17/KEY22/ PWML/PTP2/PTP2I	PE1	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG17	PES0	—	AN	LCD Segment output
	KEY22	PES0	AN	—	Touch key input
	PWML	PES0	—	CMOS	PWM complementary output
	PTP2	PES0	—	CMOS	PTM2 output
	PTP2I	PES0 IFS2	ST	—	PTM2 capture input
PE2/SEG18/STP/STPI/ KEY23/TX0/PTCK2	PE2	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG18	PES1	—	AN	LCD Segment output
	STP	PES1	—	CMOS	STM output
	STPI	PES1 IFS2	ST	—	STM capture input
	KEY23	PES1	AN	—	Touch key input
	TX0	PES1	—	CMOS	UART0 serial data output
	PTCK2	PES1	ST	—	PTM2 clock/capture input
PE3/SEG19/STPB/STPI/ KEY24/$\overline{SCS}$/RX0/TX0	PE3	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG19	PES1	—	AN	LCD Segment output
	STPB	PES1	—	CMOS	STM inverted output
	STPI	PES1 IFS2	ST	—	STM capture input
	KEY24	PES1	AN	—	Touch key input
	$\overline{SCS}$	PES1 IFS4	ST	CMOS	SIM SPI slave select pin
	RX0/TX0	PES1 IFS1	ST	CMOS	UART0 serial data input in full-duplex communication or UART0 serial data input / output in Single Wire Mode communication
PE4/SEG20/SCK/SCL/SDO/ TX2/IICSDA/PTP2I/PTP0	PE4	PEPU PES2	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG20	PES2	—	AN	LCD Segment output
	SCK	PES2 IFS0	ST	CMOS	SIM SPI serial clock
	SCL	PES2 IFS0	ST	NMOS	SIM I^2C clock line
	SDO	PES2	—	CMOS	SIM SPI serial data output
	TX2	PES2	—	CMOS	UART2 serial data output
	IICSDA	PES2 IFS3	ST	NMOS	Independent I^2C data line
	PTP2I	PES2 IFS2	ST	—	PTM2 capture input
	PTP0	PES2	—	CMOS	PTM0 output

Pin Name	Function	OPT	I/T	O/T	Description
PE5/SEG21/CTCK1/SDI/SDA/RX2/TX2/IICSCL/CTP0	PE5	PEPU PES2	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG21	PES2	—	AN	LCD Segment output
	CTCK1	PES2	ST	—	CTM1 clock input
	SDI	PES2 IFS0	ST	—	SIM SPI serial data input
	SDA	PES2 IFS0	ST	NMOS	SIM I²C data line
	RX2/TX2	PES2 IFS2	ST	CMOS	UART2 serial data input in full-duplex communication or UART2 serial data input/output in Single Wire Mode communication
	IICSCL	PES2 IFS3	ST	NMOS	Independent I²C clock line
	CTP0	PES2	—	CMOS	CTM0 output
PE6/SEG22/CTP1/SDO/SCK/SCL/TX1	PE6	PEPU PES3	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG22	PES3	—	AN	LCD Segment output
	CTP1	PES3	—	CMOS	CTM1 output
	SDO	PES3	—	CMOS	SIM SPI serial data output
	SCK	PES3 IFS0	ST	CMOS	SIM SPI serial clock
	SCL	PES3 IFS0	ST	NMOS	SIM I²C clock line
	TX1	PES3	—	CMOS	UART1 serial data output
PE7/SEG23/CTP1B/CTCK0/INT1/$\overline{SCS}$/RX1/TX1/STP	PE7	PEPU PES3	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG23	PES3	—	AN	LCD Segment output
	CTP1B	PES3	—	CMOS	CTM1 inverted output
	CTCK0	PES3 IFS2	ST	—	CTM0 clock input
	INT1	PES3 INTEG INTC0 IFS4	ST	—	External interrupt input
	$\overline{SCS}$	PES3 IFS4	ST	CMOS	SIM SPI slave select pin
	RX1/TX1	PES3 IFS1	ST	CMOS	UART1 serial data input in full-duplex communication or UART1 serial data input / output in Single Wire Mode communication
	STP	PES3	—	CMOS	STM output
PF0/SEG24/CTP0B/SCK/SCL/SDO/TX0/IICSDA	PF0	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG24	PFS0	—	AN	LCD Segment output
	CTP0B	PFS0	—	CMOS	CTM0 inverted output
	SCK	PFS0 IFS0	ST	CMOS	SIM SPI serial clock
	SCL	PFS0 IFS0	ST	NMOS	SIM I²C clock line
	SDO	PFS0	—	CMOS	SIM SPI serial data output
	TX0	PFS0	—	CMOS	UART0 serial data output
	IICSDA	PFS0 IFS3	ST	NMOS	Independent I²C data line

Pin Name	Function	OPT	I/T	O/T	Description
PF1/SEG25/CTP0/ SDI/SDA/RX0/TX0/IICSCL	PF1	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG25	PFS0	—	AN	LCD Segment output
	CTP0	PFS0	—	CMOS	CTM0 output
	SDI	PFS0 IFS0	ST	—	SIM SPI serial data input
	SDA	PFS0 IFS0	ST	NMOS	SIM I²C data line
	RX0/TX0	PFS0 IFS1	ST	CMOS	UART0 serial data input in full-duplex communication or UART0 serial data input/ output in Single Wire Mode communication
	IICSCL	PFS0 IFS3	ST	NMOS	Independent I²C clock line
PF2/SEG26/CTP0B/SDO/ SCK/SCL/TX1	PF2	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG26	PFS1	—	AN	LCD Segment output
	CTP0B	PFS1	—	CMOS	CTM0 inverted output
	SDO	PFS1	—	CMOS	SIM SPI serial data output
	SCK	PFS1 IFS0	ST	CMOS	SIM SPI serial clock
	SCL	PFS1 IFS0	ST	NMOS	SIM I²C clock line
	TX1	PFS1	—	CMOS	UART1 serial data output
PF3/SEG27/CTCK0/INT0/ $\overline{SCS}$/RX1/TX1/PTP2	PF3	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG27	PFS1	—	AN	LCD Segment output
	CTCK0	PFS1 IFS2	ST	—	CTM0 clock input
	INT0	PFS1 INTEG INTC0 IFS1	ST	—	External interrupt input
	$\overline{SCS}$	PFS1 IFS4	ST	CMOS	SIM SPI slave select pin
	RX1/TX1	PFS1 IFS1	ST	CMOS	UART1 serial data input in full-duplex communication or UART1 serial data input / output in Single Wire Mode communication
	PTP2	PFS1	—	CMOS	PTM2 output
PF4/SEG28/CTCK2/ SCK/SCL/PTP1B	PF4	PFPU PFS2	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG28	PFS2	—	AN	LCD Segment output
	CTCK2	PFS2	ST	—	CTM2 clock input
	SCK	PFS2 IFS0	ST	CMOS	SIM SPI serial clock
	SCL	PFS2 IFS0	ST	NMOS	SIM I²C clock line
	PTP1B	PFS2	—	CMOS	PTM1 inverted output

Pin Name	Function	OPT	I/T	O/T	Description
PF5/SEG29/CTP2/ SDI/SDA/PTP1/PTP2	PF5	PFPU PFS2	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG29	PFS2	—	AN	LCD Segment output
	CTP2	PFS2	—	CMOS	CTM2 output
	SDI	PFS2 IFS0	ST	—	SIM SPI serial data input
	SDA	PFS2 IFS0	ST	NMOS	SIM I²C data line
	PTP1	PFS2	—	CMOS	PTM1 output
	PTP2	PFS2	—	CMOS	PTM2 output
PF6/SEG30/CTP2B/SDO/TX2/ PWMH/IICSDA/IICSCL/PTCK1	PF6	PFPU PFS3	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG30	PFS3	—	AN	LCD Segment output
	CTP2B	PFS3	—	CMOS	CTM2 inverted output
	SDO	PFS3	—	CMOS	SIM SPI serial data output
	TX2	PFS3	—	CMOS	UART2 serial data output
	PWMH	PFS3	—	CMOS	PWM output
	IICSDA	PFS3 IFS3	ST	NMOS	Independent I²C data line
	IICSCL	PFS3 IFS3	ST	NMOS	Independent I²C clock line
	PTCK1	PFS3	ST	—	PTM1 clock/capture input
PF7/SEG31/$\overline{SCS}$/RX2/TX2/ PWML/IICSCL/IICSDA/PTP1I	PF7	PFPU PFS3	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG31	PFS3	—	AN	LCD Segment output
	$\overline{SCS}$	PFS3 IFS4	ST	CMOS	SIM SPI slave select pin
	RX2/TX2	PFS3 IFS2	ST	CMOS	UART2 serial data input in full-duplex communication or UART2 serial data input / output in Single Wire Mode communication
	PWML	PFS3	—	CMOS	PWM complementary output
	IICSCL	PFS3 IFS3	ST	NMOS	Independent I²C clock line
	IICSDA	PFS3 IFS3	ST	NMOS	Independent I²C data line
	PTP1I	PFS3	ST	—	PTM1 capture input
PG0/SEG32/COM7/RX2/TX2	PG0	PGPU PGS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG32	PGS0	—	AN	LCD Segment output
	COM7	PGS0	—	AN	LCD Common output
	RX2/TX2	PGS0 IFS2	ST	CMOS	UART2 serial data input in full-duplex communication or UART2 serial data input / output in Single Wire Mode communication
PG1/SEG33/COM6/TX2	PG1	PGPU PGS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG33	PGS0	—	AN	LCD Segment output
	COM6	PGS0	—	AN	LCD Common output
	TX2	PGS0	—	CMOS	UART2 serial data output

Pin Name	Function	OPT	I/T	O/T	Description
PG2/SEG34/COM5/IICSDA	PG2	PGPU PGS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG34	PGS0	—	AN	LCD Segment output
	COM5	PGS0	—	AN	LCD Common output
	IICSDA	PGS0 IFS3	ST	NMOS	Independent I^2C data line
PG3/SEG35/COM4/IICSCL	PG3	PGPU PGS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG35	PGS0	—	AN	LCD Segment output
	COM4	PGS0	—	AN	LCD Common output
	IICSCL	PGS0 IFS3	ST	NMOS	Independent I^2C clock line
PG4/$\overline{SCS}$/TX1/SEG36	PG4	PGPU PGS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	$\overline{SCS}$	PGS1 IFS4	ST	CMOS	SIM SPI slave select pin
	TX1	PGS1	—	CMOS	UART1 serial data output
	SEG36	PGS1	—	AN	LCD Segment output
PG5/SDO/RX1/TX1/SEG37	PG5	PGPU PGS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SDO	PGS1	—	CMOS	SIM SPI serial data output
	RX1/TX1	PGS1 IFS1	ST	CMOS	UART1 serial data input in full-duplex communication or UART1 serial data input / output in Single Wire Mode communication
	SEG37	PGS1	—	AN	LCD Segment output
PG6/SDI/SDA/RX0/TX0/SEG38	PG6	PGPU PGS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SDI	PGS1 IFS0	ST	—	SIM SPI serial data input
	SDA	PGS1 IFS0	ST	NMOS	SIM I^2C data line
	RX0/TX0	PGS1 IFS1	ST	CMOS	UART0 serial data input in full-duplex communication or UART0 serial data input / output in Single Wire Mode communication
	SEG38	PGS1	—	AN	LCD Segment output
PG7/SCK/SCL/TX0/SEG39	PG7	PGPU PGS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SCK	PGS1 IFS0	ST	CMOS	SIM SPI serial clock
	SCL	PGS1 IFS0	ST	NMOS	SIM I^2C clock line
	TX0	PGS1	—	CMOS	UART0 serial data output
	SEG39	PGS1	—	AN	LCD Segment output
PH0/COM3	PH0	PHPU PHS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	COM3	PHS0	—	AN	LCD Common output
PH1/COM2	PH1	PHPU PHS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	COM2	PHS0	—	AN	LCD Common output
PH2/COM1	PH2	PHPU PHS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	COM1	PHS0	—	AN	LCD Common output

Pin Name	Function	OPT	I/T	O/T	Description
PH3/COM0	PH3	PHPU PHS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	COM0	PHS0	—	AN	LCD Common output
V1	V1	—	PWR	AN	LCD voltage pump
VMAX	VMAX	—	PWR	—	LCD maximum voltage, should be connected to VDD or V1
PLCD	PLCD	—	PWR	AN	LCD power supply
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

Legend: I/T: Input type;
O/T: Output type;
PWR: Power;
OPT: Optional by register option;
CMOS: CMOS output;
NMOS: NMOS output;
ST: Schmitt Trigger input;
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_{OH} Total ..... -80mA

I_{OL} Total 80mA

Total Power Dissipation 500mW

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

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	f_{SYS}=8MHz	1.8	—	5.5	V
		f_{SYS}=12MHz	2.7	—	5.5	
		f_{SYS}=16MHz	3.3	—	5.5	
	Operating Voltage – LXT	f_{SYS}=32.768kHz	1.8	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32kHz	1.8	—	5.5	V

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	1.8V	WDT off	—	0.3	0.7	2.4	μA
		2.2V		—	0.3	0.8	2.6	
		3V		—	0.4	0.9	3.0	
		5V		—	0.5	1.1	4.0	
		1.8V	WDT on	—	1.2	1.6	3.1	μA
		2.2V		—	1.3	1.8	3.4	
		3V		—	1.7	2.1	4.0	
		5V		—	3.0	3.5	6.2	
	IDLE0 Mode – LIRC	1.8V	f_{SUB} on	—	1.3	1.7	3.3	μA
		2.2V		—	1.4	1.9	3.6	
		3V		—	1.8	2.2	4.2	
		5V		—	3.3	3.9	6.6	
	IDLE0 Mode – LXT	1.8V	f_{SUB} on	—	2.3	2.7	4.3	μA
		2.2V		—	2.4	2.9	4.6	
		3V		—	2.8	3.2	5.2	
		5V		—	4.3	4.9	7.6	
	IDLE1 Mode – HIRC	1.8V	f_{SUB} on, f_{SYS}=8MHz	—	220	250	270	μA
		2.2V		—	250	280	300	
		3V		—	380	410	440	
		5V		—	720	770	820	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	360	390	420	μA
		3V		—	540	580	610	
		5V		—	1070	1130	1180	
		3.3V	f_{SUB} on, f_{SYS}=16MHz	—	750	800	860	μA
		5V		—	1450	1540	1630	

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

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

Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC (LVR Enable)	1.8V	f_{SYS}=32kHz	—	19	23	μA
		2.2V		—	21	26	
		3V		—	22	27	
		5V		—	24	30	
	SLOW Mode – LXT (LVR Enable)	1.8V	f_{SYS}=32.768kHz	—	20	24	μA
		2.2V		—	22	27	
		3V		—	23	28	
		5V		—	25	31	
	FAST Mode – HIRC	1.8V	f_{SYS}=8MHz	—	0.36	0.40	mA
		2.2V		—	0.40	0.45	
		3V		—	0.57	0.64	
		5V		—	1.02	1.10	
		2.7V	f_{SYS}=12MHz	—	0.73	0.80	mA
		3V		—	0.82	0.92	
		5V		—	1.48	1.65	
		3.3V	f_{SYS}=16MHz	—	1.13	1.25	mA
		5V		—	2.03	2.25	

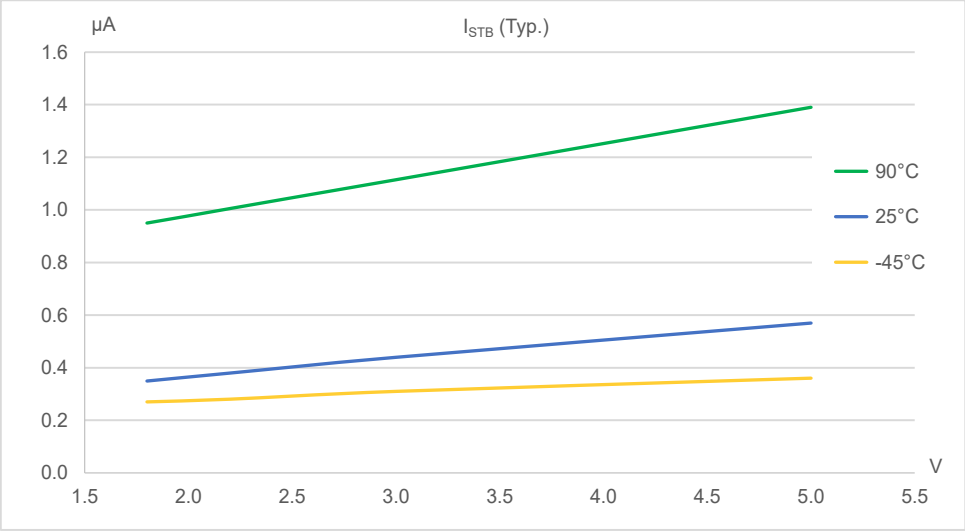
Note: When using the characteristic table data, the following notes should be taken into consideration:

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.

Operating Current Characteristic Curves

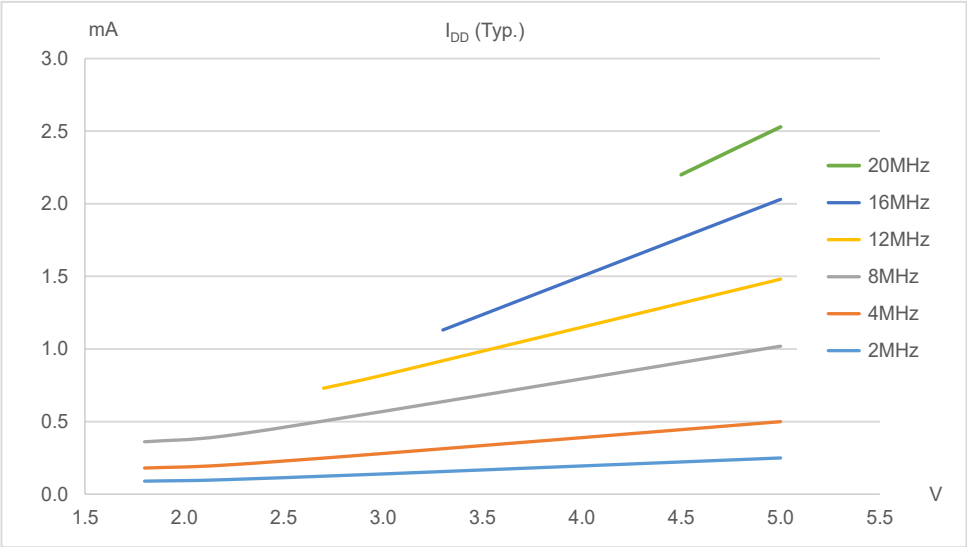
The following diagrams show the MCU operating current voltage/current relationship using different oscillators at different frequencies.

MCU SLEEP Mode Standby Current – WDT Off



Note: This I_{STB} (Typ.) curve is for reference only.

MCU FAST Mode Operating Current – HIRC



Note: This I_{DD} (Typ.) curve is for reference only.

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}	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	-3.5%	8	+3.5%	
			-40°C~85°C	-5%	8	+5%	
		1.8V~5.5V	25°C	-10%	8	+5%	
			-40°C~85°C	-15%	8	+10%	
		2.7V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	
	12MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	12	+1%	MHz
			-40°C~85°C	-2%	12	+2%	
		2.7V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C~85°C	-3%	12	+3%	
	16MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	16	+1%	MHz
			-40°C~85°C	-2%	16	+2%	
		3.3V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C~85°C	-3%	16	+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 1.8V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.
3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

Internal Low Speed Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	3V	25°C	-2%	32	+2%	kHz
		2.2V~5.5V	-40°C~85°C	-7%	32	+7%	
		1.8V~5.5V	-40°C~85°C	-10%	32	+10%	
t_{START}	LIRC Start-up Time	—	-40°C~85°C	—	—	100	μs

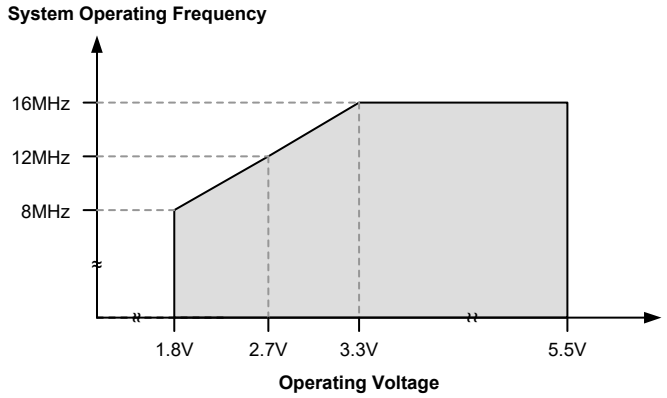
External Low Speed Crystal Oscillator Characteristics – LXT

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{LXT}	LXT Frequency	1.8V~5.5V	—	—	32768	—	Hz
t_{START}	LXT Start-up Time	3V	—	—	—	1000	ms
		5V	—	—	—	1000	
Duty Cycle	Duty Cycle	—	—	40	—	60	%
R_{NEG}	Negative Resistance	1.8V	—	3×ESR	—	—	Ω

Note: C1, C2 and R_P are external components. When the LXTSP[1:0]=00B or 01B, C1=C2=10pF, R_P=10MΩ. C_L=7pF, ESR=30kΩ. When the LXTSP[1:0]=10B or 11B, C1=C2=12pF, R_P=10MΩ. C_L=7pF, ESR=90kΩ.

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)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LXT}$	—	1024	—	t_{LXT}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time (Wake-up from Condition where f_{SYS} is on)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LXT}$ or f_{LIRC}	—	2	—	t_{SUB}
	System Speed Switch Time (FAST to Slow Mode or SLOW to FAST Mode)	—	f_{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 (LVRC/WDTC Software Reset)	—	—				
	System Reset Delay Time (Reset Source from WDT Overflow Reset)	—	—	14	16	18	
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 I/O ports except for PB1~PB3.

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+2, m+1, m]=000B (n=0~7; m=0, 3)	-0.5	-1.0	—	mA
		5V		-1.0	-2.0	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+2, m+1, m]=001B (n=0~7; m=0, 3)	-1.2	-2.5	—	mA
		5V		-2.5	-5.0	—	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH}	Source Current for I/O Ports	3V	V_{OH}=0.9V_{DD}, SLEDCn[m+2, m+1, m]=010B (n=0~7; m=0, 3)	-2.0	-4.0	—	mA
		5V		-4.0	-8.0	—	
		3V	V_{OH}=0.9V_{DD}, SLEDCn[m+2, m+1, m]=011B (n=0~7; m=0, 3)	-2.5	-5.0	—	mA
		5V		-5.5	-11.0	—	
		3V	V_{OH}=0.9V_{DD}, SLEDCn[m+2, m+1, m]=100B (n=0~7; m=0, 3)	-3.5	-7.0	—	mA
		5V		-7.0	-14.0	—	
		3V	V_{OH}=0.9V_{DD}, SLEDCn[m+2, m+1, m]=101B (n=0~7; m=0, 3)	-4.0	-8.5	—	mA
		5V		-8.5	-17.0	—	
		3V	V_{OH}=0.9V_{DD}, SLEDCn[m+2, m+1, m]=110B (n=0~7; m=0, 3)	-5	-10	—	mA
		5V		-10	-20	—	
		3V	V_{OH}=0.9V_{DD}, SLEDCn[m+2, m+1, m]=111B (n=0~7; m=0, 3)	-5.5	-11.0	—	mA
		5V		-11	-23	—	
R_{PH}	Pull-high Resistance for I/O Ports (1)	3V	LVPU=0, PxPU=FFH (Px: PA, PB, PC, PD, PE, PF, PG, PH)	20	60	100	kΩ
		5V		10	30	50	
		3V	LVPU=1, PxPU=FFH (Px: PA, PB, PC, PD, PE, PF, PG, PH)	6.67	15	23	kΩ
		5V		3.5	7.5	12	
I_{LEAK}	Input Leakage Current for I/O Ports	5V	V_{IN}=V_{DD} or V_{IN}=V_{SS}	—	—	±1	μA
t_{INT}	External Interrupt Input Pin Minimum Pulse Width	—	—	10	—	—	μs
t_{TCK}	CTMn/PTMn/STM Clock Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TPI}	PTMn/STM Capture Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
f_{TMCLK}	PTMn/STM Maximum Timer Clock Source Frequency	5V	—	—	—	1	f_{SYS}
t_{CPW}	PTMn/STM 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:

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

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

Ex1: If PTnCAPTS=0, f_{TMCLK}=8MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.25μs, 0.3μs)=0.3μs

Ex2: If PTnCAPTS=1, f_{TMCLK}=8MHz, t_{TCK}=0.3μs, then t_{CPW}=max(0.25μs, 0.3μs)=0.3μs

Ex3: If PTnCAPTS=0, f_{TMCLK}=4MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.5μs, 0.3μs)=0.5μs

For STM:

t_{CPW}=max(2×t_{TMCLK}, t_{TPI})

Ex1: If f_{TMCLK}=8MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.25μs, 0.3μs)=0.3μs

Ex2: 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 the PB1~PB3 pins.

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Power Supply – V_{DD0}	—	—	1.8	5.0	5.5	V
V_{DDIO}	Power Supply – V_{DD1}	—	—	1.8	—	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	—	mA
			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}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=000B (n=1; m=0)	-0.5	-1.0	—	mA
		5V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=000B (n=1; m=0)	-1.0	-2.0	—	mA
			V_{OH}=0.9V_{DDn}, V_{DDn}=3V, n=0~1; SLEDCn[m+2, m+1, m]=000B (n=1; m=0)	-0.3	-0.6	—	
		3V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=001B (n=1; m=0)	-1.2	-2.5	—	mA
		5V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=001B (n=1; m=0)	-2.5	-5.0	—	mA
			V_{OH}=0.9V_{DDn}, V_{DDn}=3V, n=0~1; SLEDCn[m+2, m+1, m]=001B (n=1; m=0)	-0.6	-1.3	—	
		3V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=010B (n=1; m=0)	-2.0	-4.0	—	mA
		5V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=010B (n=1; m=0)	-4.0	-8.0	—	mA
			V_{OH}=0.9V_{DDn}, V_{DDn}=3V, n=0~1; SLEDCn[m+2, m+1, m]=010B (n=1; m=0)	-1.0	-2.0	—	
		3V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=011B (n=1; m=0)	-2.5	-5.0	—	mA
		5V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=011B (n=1; m=0)	-5.5	-11.0	—	mA
			V_{OH}=0.9V_{DDn}, V_{DDn}=3V, n=0~1; SLEDCn[m+2, m+1, m]=011B (n=1; m=0)	-1.2	-2.5	—	
		3V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=100B (n=1; m=0)	-3.5	-7.0	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH}	Source Current for I/O Ports	5V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=100B (n=1; m=0)	-7	-14	—	mA
			V_{OH}=0.9V_{DDn}, V_{DDn}=3V, n=0~1; SLEDCn[m+2, m+1, m]=100B (n=1; m=0)	-1.8	-3.5	—	
		3V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=101B (n=1; m=0)	-4.0	-8.5	—	mA
		5V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=101B (n=1; m=0)	-8.5	-17.0	—	mA
			V_{OH}=0.9V_{DDn}, V_{DDn}=3V, n=0~1; SLEDCn[m+2, m+1, m]=101B (n=1; m=0)	-2.0	-4.0	—	
		3V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=110B (n=1; m=0)	-5	-10	—	mA
		5V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=110B (n=1; m=0)	-10	-20	—	mA
			V_{OH}=0.9V_{DDn}, V_{DDn}=3V, n=0~1; SLEDCn[m+2, m+1, m]=110B (n=1; m=0)	-2.5	-5.0	—	
		3V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=111B (n=1; m=0)	-5.5	-11.0	—	mA
		5V	V_{OH}=0.9V_{DDn}, V_{DDn}=V_{DD}, n=0~1; SLEDCn[m+2, m+1, m]=111B (n=1; m=0)	-11	-23	—	mA
			V_{OH}=0.9V_{DDn}, V_{DDn}=3V, n=0~1; SLEDCn[m+2, m+1, m]=111B (n=1; m=0)	-2.8	-5.5	—	
R_{PH}	Pull-high Resistance for I/O Ports (1)	3V	V_{DDn}=V_{DD}, n=0~1; LVPU=0	20	60	100	kΩ
		5V	V_{DDn}=V_{DD}, n=0~1; LVPU=0	10	30	50	
			V_{DDn}=3V, n=0~1; LVPU=0	36	110	180	
		3V	V_{DDn}=V_{DD}, n=0~1; LVPU=1	6.67	15	23	
		5V	V_{DDn}=V_{DD}, n=0~1; LVPU=1	3.5	7.5	12	
			V_{DDn}=3V, n=0~1; LVPU=1	9	27.5	45	
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 the V_{DDn} is used as the I/O power, a 0.1μF bypass capacitor should be added close to the VDDn (VDD or VDDIO) pin.

Memory Electrical Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Flash Program Memory							
t_{FWR}	Write Time	—	FWERTS=0	—	2.2	2.7	ms
		—	FWERTS=1	—	3.0	3.6	
t_{FER}	Erase Time	—	FWERTS=0	—	3.2	3.9	ms
		—	FWERTS=1	—	3.7	4.5	
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	Data Retention Time	—	Ta=25°C	—	40	—	Year
t_{ACTV}	ROM Activation Time – Wake-up from Power Down Mode	—	—	32	—	64	μs
Data EEPROM Memory							
t_{EERD}	Read Time	—	—	—	—	4	t_{SYS}
t_{EEWR}	Write Time (Byte Mode)	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	
	Write Time (Page Mode)	—	EWERTS=0	—	2.2	2.7	
		—	EWERTS=1	—	3.0	3.6	
t_{EEER}	Erase Time	—	EWERTS=0	—	3.2	3.9	ms
		—	EWERTS=1	—	3.7	4.5	
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

Note: 1. The ROM activation time t_{ACTV} should be added when calculating the total system start-up time of a wake-up from the power down mode.

2. "E/W" means Erase/Write times.

LVR & LVD Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	1.8	—	5.5	V
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 1.7V	-5%	1.7	+5%	V
		—	LVR enable, voltage select 1.9V	-5%	1.9	+5%	
		—	LVR enable, voltage select 2.55V	-3%	2.55	+3%	
		—	LVR enable, voltage select 3.15V	-3%	3.15	+3%	
		—	LVR enable, voltage select 3.8V	-3%	3.8	+3%	
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 1.8V	-5%	1.8	+5%	V
		—	LVD enable, voltage select 2.0V		2.0		
		—	LVD enable, voltage select 2.4V		2.4		
		—	LVD enable, voltage select 2.7V		2.7		
		—	LVD enable, voltage select 3.0V		3.0		
		—	LVD enable, voltage select 3.3V		3.3		
		—	LVD enable, voltage select 3.6V		3.6		
		—	LVD enable, voltage select 4.0V		4.0		

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{LVRLVD}	Operating Current	3V	LVD enable, LVR enable, V_{LVR}=1.9V, V_{LVD}=2.0V	—	—	10	μA
		5V	LVD enable, LVR enable, V_{LVR}=1.9V, V_{LVD}=2.0V	—	10	15	
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_{LVDS}	LVDO Stable Time	—	For LVR enable, LVD off → on	—	—	18	μs
			For LVR disable, LVD off → on	—	—	150	
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	
I_{LVR}	Additional Current for LVR Enable	5V	LVD disable	—	—	8	μA
I_{LVD}	Additional Current for LVD Enable	5V	LVR disable	—	—	8	μA

Internal Reference Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{BG}	Bandgap Reference Voltage	—	—	-20%	0.93	+20%	V
t_{BGS}	V_{BG} Turn-on Stable Time	—	No load	—	—	50	μs
I_{BG}	Additional Current for Bandgap Reference Enable	—	VBGEN=1, LVR/LVD disable	—	—	2	μA

Note: The V_{BG} voltage is used as the A/D converter PGA input.

A/D Converter Electrical Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	1.8	—	V_{DD}	V
N_R	Resolution	—	—	—	—	12	Bit
DNL	Differential Non-linearity	1.8V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V_{REF}=V_{DD}, t_{ADCK}=2.0μs	-3	—	3	LSB
		2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V_{REF}=V_{DD}, t_{ADCK}=0.5μs				
		3V					
		5V					
		1.8V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V_{REF}=V_{DD}, t_{ADCK}=10μs				
		3V					
		5V					

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
INL	Integral Non-linearity	1.8V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, $V_{REF}=V_{DD}$, t_{ADCK}=2.0μs	-4	—	4	LSB
		2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, $V_{REF}=V_{DD}$, t_{ADCK}=0.5μs				
		3V					
		5V					
		1.8V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, $V_{REF}=V_{DD}$, t_{ADCK}=10μs				
		3V					
		5V					
I_{ADC}	Additional Current for A/D Converter Enable	1.8V	No load, t_{ADCK}=2.0μs	—	280	400	μA
		3V	No load, t_{ADCK}=0.5μs	—	450	600	
		5V	No load, t_{ADCK}=0.5μs	—	850	1000	
t_{ADCK}	Clock Period	1.8V≤V_{DD}<2.0V	AN≠Temperature Sensor	2.0	—	10.0	μs
		2.0V≤V_{DD}≤5.5V		0.5	—	10.0	
		—	AN=Temperature Sensor	4	8	10	
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t_{ADC}	Conversion Time (Including A/D Sample and Hold Time)	—	—	—	16	—	t_{ADCK}
I_{PGA}	Additional Current for PGA Enable	2.2V	No load, PGAIS[1:0]=01B, PGAGS[1:0]=01B	—	250	500	μA
		3V		—	300	600	
		5V		—	400	700	
V_{OR}	PGA Maximum Output Voltage Range	2.2V	—	V_{SS} +0.1	—	V_{DD} -0.1	V
		3V		V_{SS} +0.1	—	V_{DD} -0.1	
		5V		V_{SS} +0.1	—	V_{DD} -0.1	
V_{VR}	Fix Voltage Output of PGA	2.2V~5.5V	Ta=25°C, $V_{RI}=V_{BG}$ (PGAIS[1:0]=01B)	-2.5%	1.6	+2.5%	V
		3.2V~5.5V			3		
		4.2V~5.5V			4		
		2.2V~5.5V	Ta=-40°C~85°C, $V_{RI}=V_{BG}$ (PGAIS[1:0]=01B)	-10%	1.6	+10%	V
		3.2V~5.5V			3		
		4.2V~5.5V			4		
V_{IR}	PGA Input Voltage Range	3V	Gain=1, PGAIS[1:0]=00B, Relative gain, Gain error<±5%	V_{SS} +0.1	—	V_{DD} -1.4	V
		5V		V_{SS} +0.1	—	V_{DD} -1.4	

Temperature Sensor Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
I_{TS}	Additional Current for Temperature Sensor Enable	—	TSEN=1 (A/D converter and PGA not included)	—	16	30	μA
t_{TSS}	Temperature Sensor Turn-On Stable Time	—	TSEN=0→1	—	—	150	μs
T_{ACC}	Temperature Accuracy (Error) (Note)	3V	Ta=25°C	-2.0	—	2.0	°C
		2.2V~5.5V	Ta=20°C~70°C, V_{REF}=1.6V, t_{ADCK}=8μs	-5.0	—	5.0	

Note: 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. Refer to the Temperature Sensor Function section for the formula. The temperature calibration is implemented at the condition of V_{DD}=3V and Ta=25±5°C.

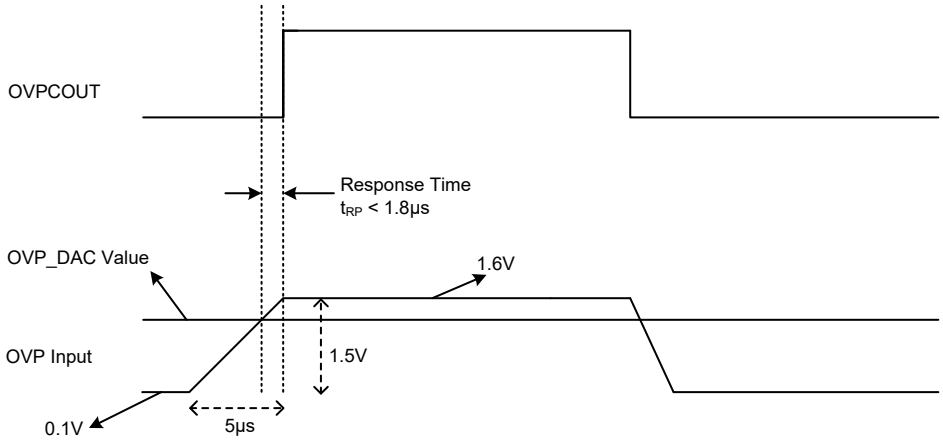
Over Voltage Protection Electrical Characteristics

Ta=-40°C~85°C

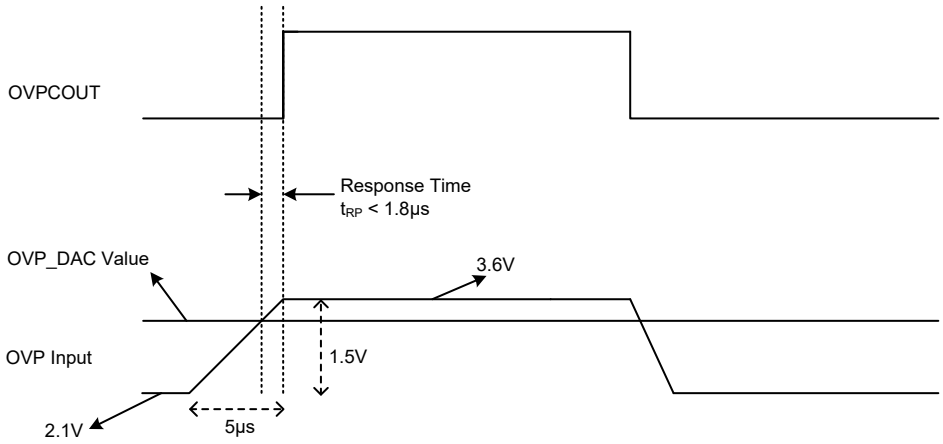
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	1.8	—	5.5	V
I_{OVP}	Operating Current	3V	OVPEN=1, DACEN=0	—	80	120	μA
		5V		—	100	140	
		3V	OVPEN=1, DACEN=1	—	300	450	
		5V		—	500	750	
V_{OS}	Input Offset Voltage	3V	With calibration	-2	—	2	mV
		5V		-2	—	2	
V_{HYS}	Hysteresis	3V	HYS[1:0]=00B	0	0	5	mV
		5V		0	0	5	
		3V	HYS[1:0]=01B	15	30	50	
		5V		15	30	50	
		3V	HYS[1:0]=10B	30	60	90	
		5V		30	60	90	
		3V	HYS[1:0]=11B	40	90	130	
		5V		40	90	130	
V_{CM}	Common Mode Voltage Range	3V	—	V_{SS}	—	V_{DD}-1	V
		5V	—	V_{SS}	—	V_{DD}-1	V
DNL	Differential Non-linearity	3V	DAC V_{REF}=V_{DD}	—	—	±1.5	LSB
		5V		—	—	±1	
INL	Integral Non-linearity	3V	DAC V_{REF}=V_{DD}	—	—	±2.0	LSB
		5V		—	—	±1.5	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{RP}	OVP Response Time	3V	OVPDA=10000000B DAC $V_{REF}=V_{DD}$ OVP Input=0.1V~1.6V(1)	—	1	1.8	μs
		5V	OVPDA=10110011B OVPDEB[2:0]=000B DAC $V_{REF}=V_{DD}$ OVP Input=2.1V~3.6V(2)	—	1	1.8	μs
		3V	With 60mV overdrive	—	—	1	μs
		5V		—	—	1	μs

Note: 1. OVP response time t_{RP} verify waveform (3V):



2. OVP response time t_{RP} verify waveform (5V):



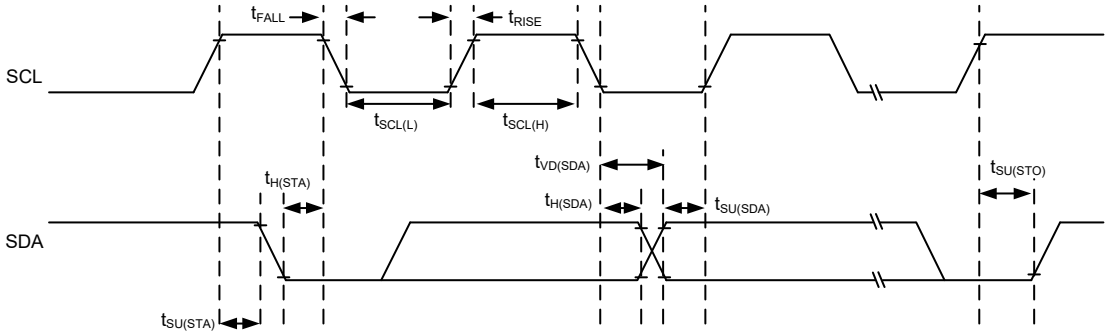
I²C Electrical Characteristics

For both the SIM I²C interface and Independent I²C interface.

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	
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 (f_{SYS}≥8MHz)	0.6	—	—	
			Fast mode (f_{SYS}<8MHz)	0.8	—	—	
$t_{SU(STO)}$	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

LCD Driver Electrical Characteristics

Ta=-40°C~85°C

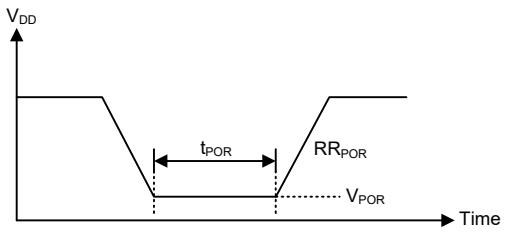
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IN}	LCD Operating Voltage	—	R type, power supply from PLCD pin, PLCD[3:0]=1xxxB	3.0	—	5.5	V
		—	C type, power supply from PLCD pin	2.0	—	3.7	
		—	C type, power supply from V1 pin	3.0	—	5.5	
		—	C type, power supply from V2 pin	1.0	—	1.8	
		—	C type, power supply from V_A	3.0	—	5.5	
		—	C type, power supply from V_C, no load	-10%	1.0	+10%	
I_{LCD}	Additional Current for LCD Enable – R Type	3V	R_T=2340kΩ, No load, V_A=V_{DD}, 1/3 bias	—	1.5	3.0	μA
		5V		—	2.5	5.0	
		3V	R_T=1170kΩ, No load, V_A=V_{PLCD}=V_{DD}, 1/3 bias & 1/4 bias	—	3	6	μA
		5V		—	5	10	
		3V	R_T=225kΩ, No load, V_A=V_{PLCD}=V_{DD}, 1/3 bias & 1/4 bias	—	16	28	μA
		5V		—	21	40	
		3V	R_T=60kΩ, No load, V_A=V_{PLCD}=V_{DD}, 1/3 bias & 1/4 bias	—	50	75	μA
		5V		—	80	120	
	Additional Current for LCD Enable – C Type	3V	No load, V_A=V1=V_{DD}, 1/3 bias, LCDPCK[2:0]=000B	—	0.6	1.2	μA
		5V		—	1	2	
		3V	No load, V_A=V1=V_{DD}, 1/3 bias, LCDPCK[2:0]=001B~111B	—	1.5	3.0	μA
		5V		—	2.4	4.8	
I_{LCDOL}	LCD Common and Segment Sink Current	3V	V_{OL}=0.1V_{DD}	210	420	—	μA
		5V		350	700	—	
I_{LCDOH}	LCD Common and Segment Source Current	3V	V_{OH}=0.9V_{DD}	-80	-160	—	μA
		5V		-180	-360	—	

Note: The V_{DD} value must be greater than or equal to the V_1 or V_A voltage, refer to the LCD Driver chapter for more details.

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

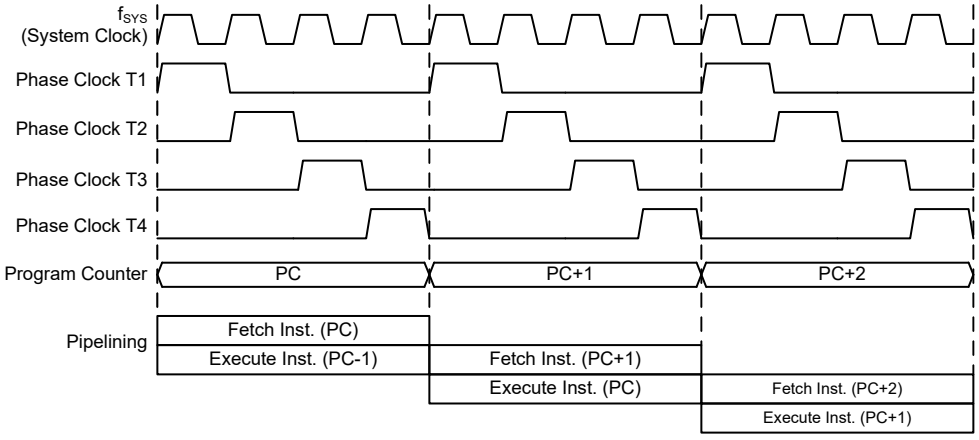


System Architecture

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

Clocking and Pipelining

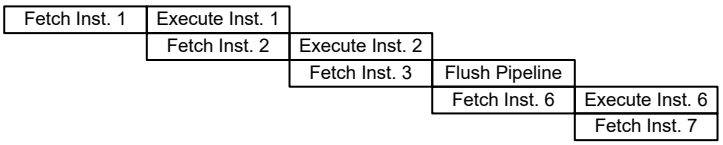
The main system clock, derived from either an HIRC, LIRC or LXT oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out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



System Clocking and Pipelining

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

```
1          MOV A,[12H]
2          CALL DELAY
3          CPL [12H]
4          :
5          :
6  DELAY:  NOP
```



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demand a jump to a non-consecutive Program Memory address. For the device with a Program Memory capacity in excess of 8K words, the Program Memory 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.

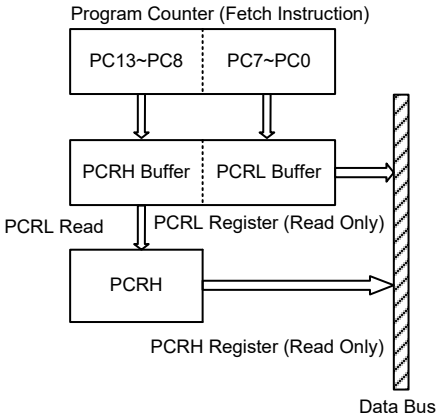
The following example shows how to read the current program execution address. When the current program execution address is 123H, the steps to execute the instructions are as follows:

(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"

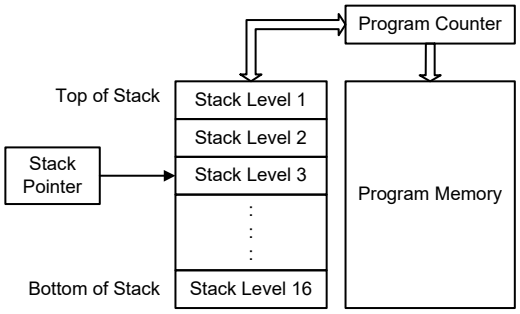
Bit 5~0 **D13~D8**: Program Counter Read High byte register bit 5 ~ bit 0

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

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

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



• **STKPTR Register**

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

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

Bit 6~4 Unimplemented, read as "0"

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

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

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

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

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

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

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

Arithmetic and Logic Unit – ALU

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

- Arithmetic operations:

  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,

  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:

  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,

  LAND, LANDM, LOR, LORM, LXOR, 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,

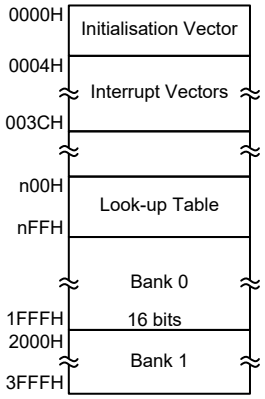
  LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

Flash Program Memory

The Program Memory is the location where the user code or program is stored. For the 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 for the device. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer registers.



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

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

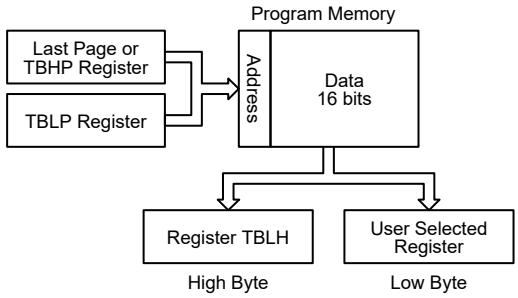


Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is "1F00H" which is located in ROM Bank 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 pointed by TBHP and TBLP if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

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

Table Read Program Example

```
rombank 1 code1
ds .section 'data'
tempreg1 db?      ; temporary register #1
tempreg2 db?      ; temporary register #2
code0 .section 'code'
mov a,06h         ; initialise table pointer - note that this address is referenced
mov tblp,a        ; to the last page or the page that tbhp pointed
mov a,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 tempreg2, the value "00H" will be
                  ; transferred to the high byte register TBLH
:
:
```

```
Code1 .section 'code'
org 1F00h         ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

Unique Identifier – UID

The device contains a 128-bit unique ID. It is unchangeable and determined by MCU manufacturer. The UID format is shown below:

Mapped Address in Program Memory Last Page		UID No.
0xE8	Low byte	UID0
	High byte	UID1
0xE9	Low byte	UID2
	High byte	UID3
0xEA	Low byte	UID4
	High byte	UID5
0xEB	Low byte	UID6
	High byte	UID7
0xEC	Low byte	UID8
	High byte	UID9
0xED	Low byte	UID10
	High byte	UID11
0xEE	Low byte	UID12
	High byte	UID13
0xEF	Low byte	CRC16[7:0]
	High byte	CRC16[15:8]

The UID is located at the Option Memory addresses 28H~2FH which will be mapped to the Program Memory last page addresses E8H~EFH. The UID can be read from the Program Memory last page using the table read instruction when the Option Memory mapping function is enabled via the ORMC register. For more details, refer to the "Option Memory Mapping Register – ORMC" in the Special Function Register Description section.

CRC Description

Calculation sequence: UID0→UID1→UID2→…→UID11→UID12→UID13.

CRC Specification:

Generating polynomial: 1021H, $X^{16}+X^{12}+X^5+1$

Polynomial=1021H

Initial Value=0000H

Final Xor Value=0000H

Input reflected=No

Output reflected=No

CRC Calculation Example:

Write 4 bytes input data sequentially and the CRC checksum are sequentially listed in the following table.

CRC Data Input / CRC Polynomial	78H→56H→34H→12H
1021H ($X^{16}+X^{12}+X^5+1$)	FF9FH→BBC3H→A367H→D0FAH

In Circuit Programming – ICP

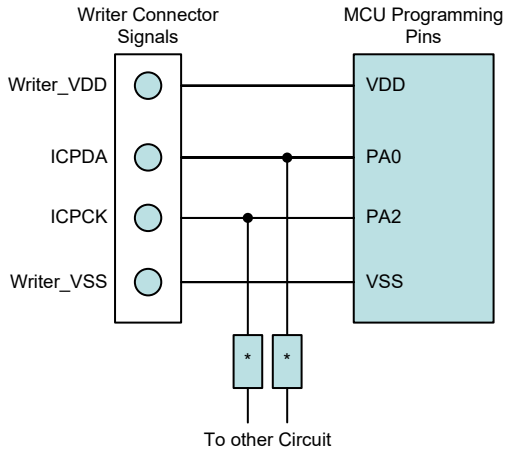
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

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

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

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



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

On-Chip Debug Support – OCDS

There is an EV chip named BS67V360CA which is used to emulate the BS67F360CA device. The EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the 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 device for debugging, other pin 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 OCDS 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 32 words. Note that the Erase operation should be executed before the Write operation is executed.

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

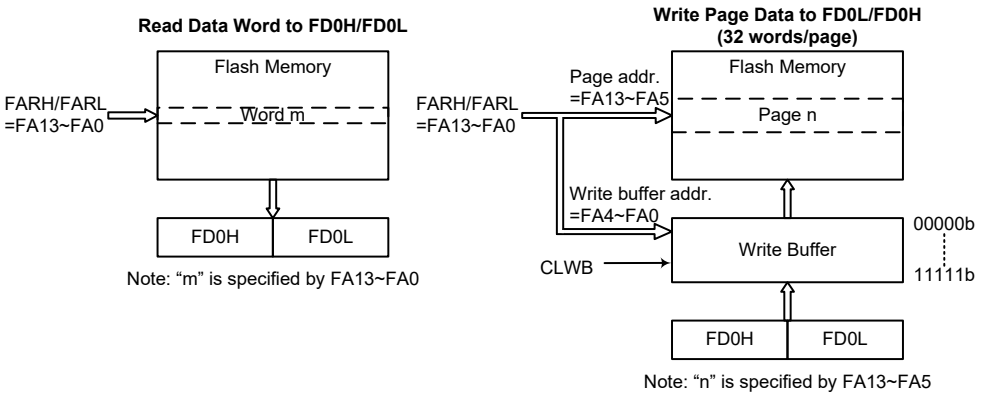
The Read operation can be carried out by executing a specific read procedure. The FRDEN bit is used to enable the read function and the FRD bit is used to initiate the read process by application programs and then indicate the read operation status. When the read process is finished, this bit will be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

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

IAP Operation Format

Page	FARH	FARL[7:5]	FARL[4:0]
0	0000 0000	000	Tag Address
1	0000 0000	001	
2	0000 0000	010	
3	0000 0000	011	
4	0000 0000	100	
: :	: :	: :	
510	0011 1111	110	
511	0011 1111	111	

Page Number and Address Selection



Flash Memory IAP Read/Write Structure

Write Buffer

The write buffer is used to store the written data temporarily when executing the write operation. The Write Buffer can be filled with written data after the Flash Memory Erase/Write Function has been successfully enabled by executing the Flash Memory Erase/Write Function Enable procedure. The write buffer can be cleared by configuring the CLWB bit in the FC2 register. The CLWB bit can be set high to enable the Clear Write Buffer procedure. When the procedure is finished this bit will be cleared to 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 32 words corresponding to a page. The write buffer address is mapped to a specific Flash memory page specified by the memory address bits, FA13~FA5. The data written into the FD0L and FD0H registers will be loaded into the write buffer. When data is written into the high byte data register, FD0H, it will result in the data stored in the high and low byte data registers both being written into the write buffer. It will also cause the Flash memory address to be incremented by one, after which the new address will be loaded into the FARH and FARL address registers. When the Flash memory address reaches the page boundary, 11111b of a page with 32 words, the address will now not be incremented but 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 16-bit data registers and three control registers, which are all located in Sector 0. 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 these registers are all located in Sector 0, they can be addressed directly using instructions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	FWERTS	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	—	FA13	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Register List

- **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	—	—	—	—	—	—	FWERTS	CLWB
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

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

Bit 0 **CLWB**: Flash memory Write Buffer Clear control
0: Do not initiate a Write Buffer Clear process or indicating that a Write Buffer Clear process has completed
1: Initiate Write Buffer Clear process
This bit is set by software and cleared 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

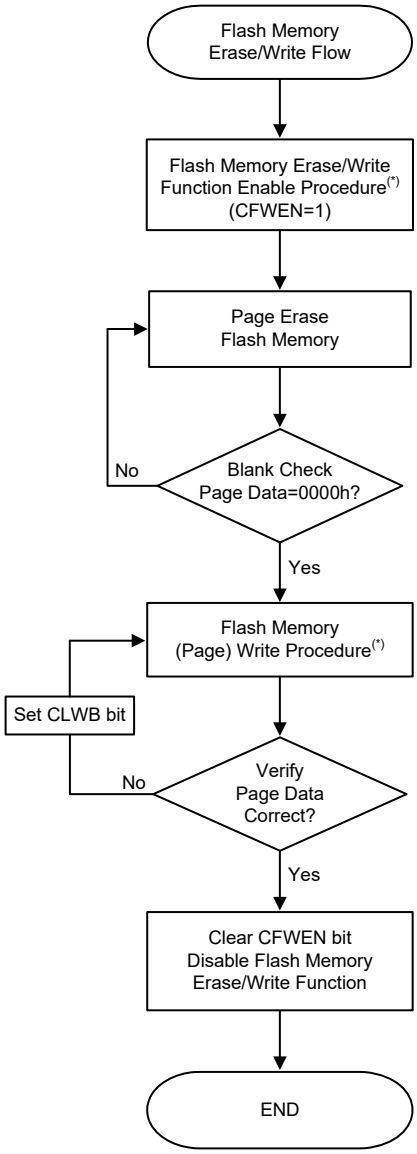
Flash Memory Erase/Write Flow

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, tag address and then erase this page.

   For a page erase operation, set the FARL and FARH registers to specify the start address of the erase page, then write dummy data into the FD0H register to tag address. The current address will be internally incremented by one after each dummy data is written into the FD0H register. When the address reaches the page boundary, 11111b, the address will not be further incremented but stop at the last address of the page. Note that the write operation to the FD0H register is used to tag address, it must be implemented to determine which addresses to be erased.
3. Execute a Blank Check operation to ensure whether the page erase operation is successful or not. The "TABRD" instruction should be executed to read the Flash memory contents and to check if the contents is 0000h or not. If the Flash memory page erase operation fails, users should go back to Step 2 and execute the page erase operation again.
4. Write data into the specific page. Refer to the "Flash Memory Write Procedure" for details.
5. Execute the "TABRD" instruction to read the Flash memory contents and check if the written data is correct or not. If the data read from the Flash memory is different from the written data, it means that the page write operation has failed. The CLWB bit should be set high to clear the write buffer and then write the data into the specific page again if the write operation has failed.
6. Clear the CFWEN bit to disable the Flash Memory Erase/Write function enable mode if the current page Erase and Write operations are complete if no more pages need to be erased or written.



Flash Memory Erase/Write Flow

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

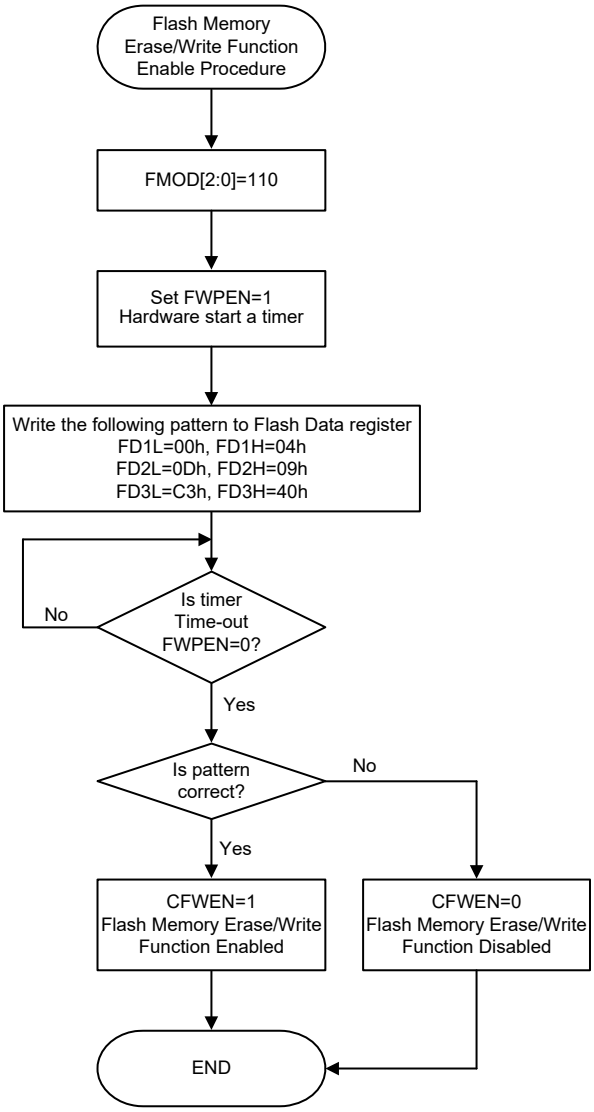
Flash Memory Erase/Write Function Enable Procedure

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

Flash Memory Erase/Write Function Enable Procedure Description

1. Write data "110" to the FMOD [2:0] bits in the FC0 register to select the Flash Memory Erase/ Write Function Enable Mode.
2. Set the FWPEN bit in the FC0 register to "1" to activate the Flash Memory Erase/Write Function. This will also activate an internal timer.
3. Write the correct data pattern into the Flash data registers, FD1L~FD3L and FD1H~FD3H, as soon as possible after the FWPEN bit is set high. The enable Flash memory erase/write function data pattern is 00H, 0DH, C3H, 04H, 09H and 40H corresponding to the FD1L~FD3L and FD1H~FD3H registers respectively.
4. Once the timer has timed out, the FWPEN bit will automatically be cleared to 0 by hardware regardless of the input data pattern.
5. If the written data pattern is incorrect, the Flash memory erase/write function will not be enabled successfully and the above steps should be repeated. If the written data pattern is correct, the Flash memory erase/write function will be enabled successfully.
6. Once the Flash memory 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 32 words, known as a page, whose address is mapped to a specific Flash memory page specified by the memory address bits, FA13~FA5. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits, FA13~FA5, specify.

Flash Memory Consecutive Write Description

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

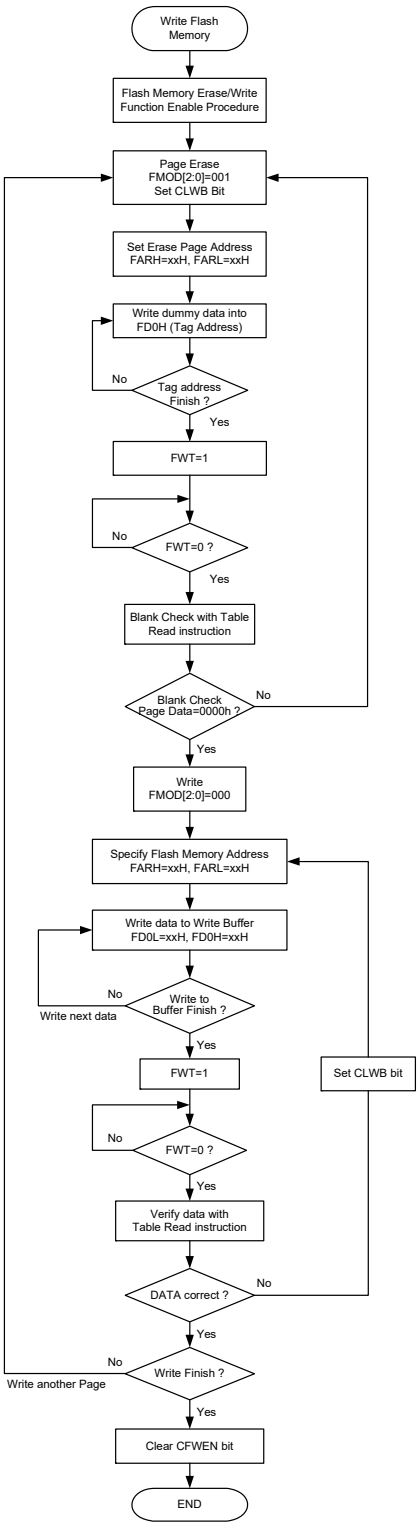
1. Activate the "Flash Memory Erase/Write function enable procedure". Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the "Flash Memory Erase/Write function enable procedure" for more details.
2. Set the FMOD field to "001" to select the erase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

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

   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to "000" to select the write operation.
5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L and FD0H registers within a page as specified by their consecutive addresses. The maximum written data number is 32 words.
6. Set the FWT bit high to write the data words from the write buffer to the Flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

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

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



Flash Memory Consecutive Write Procedure

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

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

Flash Memory Non-consecutive Write Description

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

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

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

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

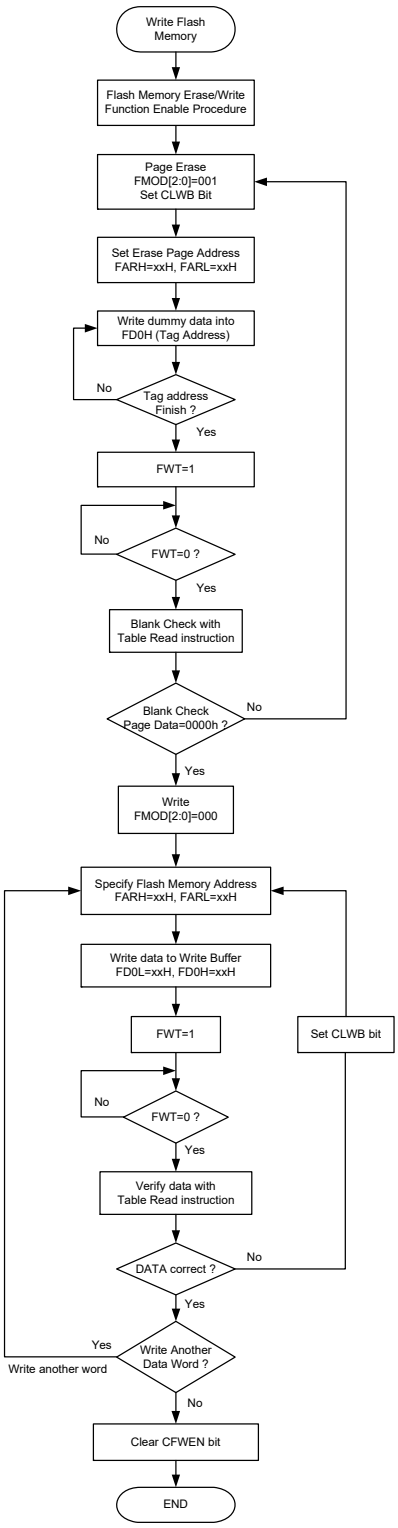
   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to "000" to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
6. Set the FWT bit high to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

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

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

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

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



Flash Memory Non-consecutive Write Procedure

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

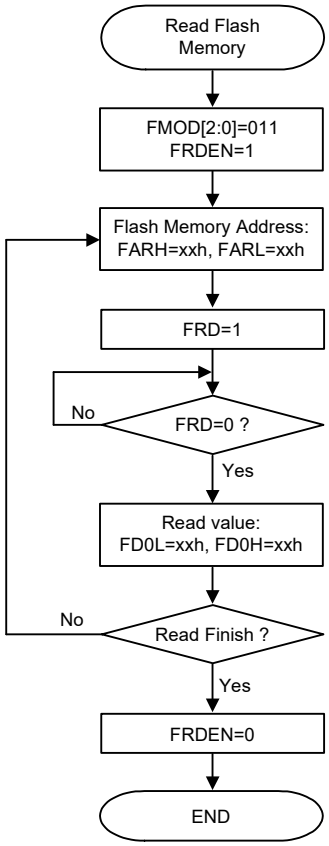
2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

Important Points to Note for Flash Memory Write Operations

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

Flash Memory Read Procedure

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



Flash Memory Read Procedure

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

2. It will take a typical time of three instruction cycles for the FRD bit state changing from high to low.

Data Memory

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

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

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

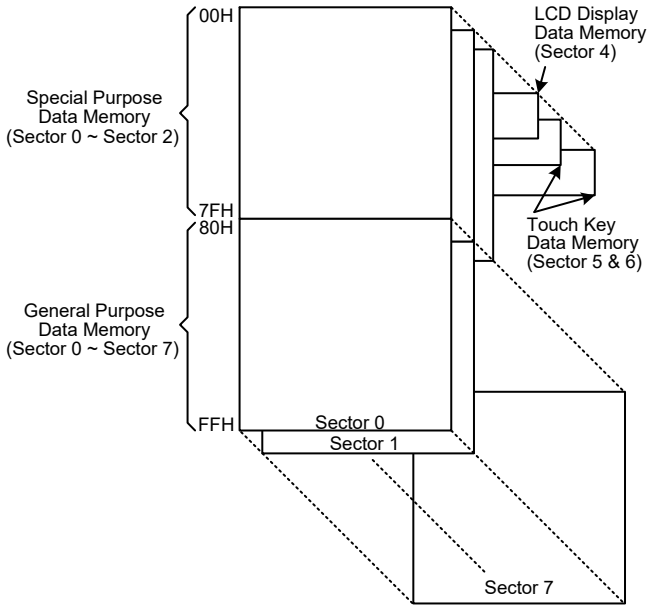
Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide Memory. Each of the Data Memory Sector is categorized into two types, the Special Purpose Data Memory and the General Purpose Data Memory. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH. The LCD Display Data Memory is located from 00H to 2BH in Sector 4. The Touch Key Data Memory is located from 00H to 3FH in Sector 5 and Sector 6.

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

Special Purpose Data Memory	General Purpose Data Memory		Touch Key Data Memory	LCD Display Data Memory
Located Sectors	Capacity	Sector: Address	Sector: Address	Sector: Address
0~2	1024×8	0: 80H~FFH 1: 80H~FFH : 7: 80H~FFH	TKRAMC=1: 5: 00H~2FH 6: 00H~2FH TKRAMC=0: 5: 00H~3FH 6: 00H~3FH	4: 00H~2BH

Data Memory Summary



Data Memory Structure

Data Memory Addressing

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

Direct Addressing can be used in all sectors using the extended 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 11 valid bits for this device, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0	Sector 1	Sector 2
00H	IAR0	TKTMR	U1SR
01H	MP0	TKC0	U1CR1
02H	IAR1	TK16DL	U1CR2
03H	MP1L	TK16DH	U1CR3
04H	MP1H	TKC1	BRDH1
05H	ACC	TKM016DL	BRDL1
06H	PCL	TKM016DH	UFCR1
07H	TBLP	TKM0ROL	TXR_RXR1
08H	TBLH	TKM0ROH	RxCNT1
09H	TBHP	TKM0C0	U2SR
0AH	STATUS	TKM0C1	U2CR1
0BH	PBP	TKM0C2	U2CR2
0CH	IAR2	TKM116DL	U2CR3
0DH	MP2L	TKM116DH	BRDH2
0EH	MP2H	TKM1ROL	BRDL2
0FH	RSTFC	TKM1ROH	UFCR2
10H	INTC0	TKM1C0	TXR_RXR2
11H	INTC1	TKM1C1	RxCNT2
12H	INTC2	TKM1C2	CPR
13H	INTC3	TKM216DL	OCVPC
14H	PA	TKM216DH	CRCCR
15H	PAC	TKM2ROL	CRCIN
16H	PAPU	TKM2ROH	CRCDL
17H	PAWU	TKM2C0	CRCDH
18H	PB	TKM2C1	IICC0
19H	PBC	TKM2C2	IICC1
1AH	PBPU	TKM316DL	IICD
1BH	INTEG	TKM316DH	IICA
1CH	SCC	TKM3ROL	IICTOC
1DH	HIRCC	TKM3ROH	
1EH	LXTC	TKM3C0	
1FH	LVRC	TKM3C1	
20H	LVDC	TKM3C2	PTM2C0
21H	TLVRC	TKM416DL	PTM2C1
22H	WDTC	TKM416DH	PTM2DL
23H		TKM4ROL	PTM2DH
24H	PC	TKM4ROH	PTM2AL
25H	PCC	TKM4C0	PTM2AH
26H	PCPU	TKM4C1	PTM2RPL
27H	PD	TKM4C2	PTM2RPH
28H	PDC	TKM516DL	
29H	PDPU	TKM516DH	
2AH	MFI0	TKM5ROL	
2BH	MFI1	TKM5ROH	
2CH	MFI2	TKM5C0	
2DH	MFI3	TKM5C1	
2EH	MFI4	TKM5C2	
2FH		PTM1C0	
30H		PTM1C1	
31H	PSC0R	PTM1DL	
32H	TB0C	PTM1DH	
33H	TB1C	PTM1AL	
34H	SIMC0	PTM1AH	
35H	SIMC1	PTM1RPL	
36H	SIMD	PTM1RPH	
37H	SIMA/SIMC2	U0SR	
38H	SIMTOC	U0CR1	
39H		U0CR2	
3AH	CTM0C0	U0CR3	
3BH	CTM0C1	BRDH0	
3CH	CTM0DL	BRDL0	
3DH	CTM0DH	UFCR0	
3EH	CTM0AL	TXR_TXR0	
3FH	CTM0AH	RxCNT0	

	Sector 0	Sector 1	Sector 2
40H		EEC	
41H	EEAL	IFS0	
42H	EEAH	IFS1	
43H	EED	IFS2	
44H	PSC1R	IFS3	
45H	LCDC0	PAS0	
46H	LCDC1	PAS1	
47H	LCDC2	PAS2	
48H	PTM0C0	PAS3	
49H	PTM0C1	PBS0	
4AH	PTM0DL	PBS1	
4BH	PTM0DH	PBS2	
4CH	PTM0AL	PBS3	
4DH	PTM0AH	PCS0	
4EH	PTM0RPL	PCS1	
4FH	PTM0RPH	PDS0	
50H	FC0	PDS1	
51H	FC1	PES0	
52H	FC2	PES1	
53H	FARL	PES2	
54H	FARH	PES3	
55H	FD0L	PFS0	
56H	FD0H	PFS1	
57H	FD1L	PFS2	
58H	FD1H	PFS3	
59H	FD2L	PGS0	
5AH	FD2H	PGS1	
5BH	FD3L	PHS0	
5CH	FD3H	IFS4	
5DH	STMC0		
5EH	STMC1	PCRL	
5FH	STMDL	PCRH	
60H	STMDH	SADC0	
61H	STMAL	SADC1	
62H	STMAH	SADC2	
63H	STMRP	SADC3	
64H	CTM1C0	SADC4	
65H	CTM1C1	SADOL	
66H	CTM1DL	SADOH	
67H	CTM1DH	SADO1BL	
68H	CTM1AL	SADO1BH	
69H	CTM1AH	SADO2BL	
6AH	CTM2C0	SADO2BH	
6BH	CTM2C1	LEBC	
6CH	CTM2DL	ATAC1C	
6DH	CTM2DH	ATAC2C	
6EH	CTM2AL	ATADT	
6FH	CTM2AH	TCRL	
70H	PE	TCRH	
71H	PEC	TCRC	
72H	PEPU	SLEDC0	
73H	PF	SLEDC1	
74H	PFC	SLEDC2	
75H	PFPU	SLEDC3	
76H	PG	SLEDC4	
77H	PGC	SLEDC5	
78H	PGPU	SLEDC6	
79H	PH	SLEDC7	
7AH	PHC	LVPUC	
7BH	PHPU	IECC	
7CH	STKPTR	OVPC0	
7DH	PMPS	OVPC1	
7EH		OVPC2	
7FH	ORMC	OVPDA	

: Unused, read as 00H

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

Example 1

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

```
    jmp loop
continue:
```

Example 2

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

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

Direct Addressing Program Example using extended instructions

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

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

Program Memory Bank Pointer – PBP

For the device the Program Memory is divided into 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-consecutive Program Memory address which is located in a certain bank selected by the program memory bank pointer bits will occur.

• **PBP Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **PBP0**: Program Memory Bank selection bit
0: Bank 0
1: Bank 1

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Option Memory Mapping Register – ORMC

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

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

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

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

• **ORMC Register**

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

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

Status Register – STATUS

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

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

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

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

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

• **STATUS Register**

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

"x": unknown

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

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

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

For SBC/ SBCM/ LSBC/ LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation zero flag. For other instructions, the CZ flag will not be affected.

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

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

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

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

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

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

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

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 1024×8 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 either the byte mode or page mode determined by the mode selection bit, MODE, in the control register, EEC.

EEPROM Registers

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

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

EEPROM Register List

- **EEAL Register**

Bit	7	6	5	4	3	2	1	0
Name	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **EEAL7~EEAL0**: Data EEPROM address low byte register bit 7 ~ bit 0
Data EEPROM address bit 7 ~ bit 0

• **EEAH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	EEAH1	EEAH0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **EEAH1~EEAH0**: Data EEPROM address high byte register bit 1 ~ bit 0
Data EEPROM address bit 9 ~ bit 8

• **EED Register**

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

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

• **EEC Register**

Bit	7	6	5	4	3	2	1	0
Name	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **EWERTS**: Data EEPROM Erase time and Write time selection
0: Erase time is 3.2ms (t_{EEER})/Write time is 2.2ms (t_{EEWR})
1: Erase time is 3.7ms (t_{EEER})/Write time is 3.0ms (t_{EEWR})

Bit 6 **EREN**: Data EEPROM erase enable
0: Disable
1: Enable
This bit is used to enable Data EEPROM erase function and must be set high before Data EEPROM erase operations are carried out. This bit will be automatically reset to zero by hardware after the erase cycle has finished. Clearing this bit to zero will inhibit data EEPROM erase operations.

Bit 5 **ER**: Data EEPROM erase control
0: Erase cycle has finished
1: Activate an erase cycle
This is the Data EEPROM Erase Control Bit. When this bit is set high by the application program, an erase cycle will be activated. This bit will be automatically reset to zero by hardware after the erase cycle has finished. Setting this bit high will have no effect if the EREN has not first been set high.

Bit 4 **MODE**: Data EEPROM operation mode selection
0: Byte operation mode
1: Page operation mode
This is the EEPROM operation mode selection bit. When the bit is set high by the application program, the Page write, erase or read function will be selected. Otherwise, the byte write or read function will be selected. The EEPROM page buffer size is 16 bytes.

Bit 3 **WREN**: Data EEPROM write enable
0: Disable
1: Enable
This is the Data EEPROM Write Enable Bit, which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations. Note that the WREN bit will automatically be cleared to zero after the write operation is finished.

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

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

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

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

Read Operation from the EEPROM

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

Byte Read Mode

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

Page Read Mode

The EEPROM page read operation can be executed when the mode selection bit, MODE, is set high. The page size can be up to 16 bytes for the page read operation. For a page read operation the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM page read operation. Note that setting the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the current byte read cycle terminates, the RD bit will automatically be cleared to zero indicating that the EEPROM data can be read from the EED register and then the current address will be incremented by one by hardware. The data which is stored in the next EEPROM address can continuously be read when

the RD bit is again set high without reconfiguring the EEPROM address and RDEN control bit. The application program can poll the RD bit to determine when the data is valid for reading.

The EEPROM address higher 6 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page read operation mode the lower 4-bit address value will automatically be incremented by one. However, the higher 6-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”.

Page Erase Operation to the EEPROM

The EEPROM page erase operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page erase. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM erase enable control bit, namely EREN, is changed from “1” to “0”, the internal page buffer will also be cleared. Note that when the EREN bit is changed from “0” to “1”, the internal page buffer will not be cleared. The EEPROM address higher 6 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page erase operation mode the lower 4-bit address value will automatically be incremented by one after each dummy data byte is written into the EED register. However, the higher 6-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”.

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

Note: The above steps must be executed sequentially to successfully complete the page erase operation, refer to the corresponding programming example.

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

Write Operation to the EEPROM

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

Byte Write Mode

The EEPROM byte write operation can be executed when the mode selection bit, MODE, is cleared to zero. For byte write operations the desired EEPROM address should first be placed in the EEAH and EEAL registers, then the data to be written should be placed in the EED register. To write data to the EEPROM, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

Note: The above steps must be executed sequentially to successfully complete the byte write operation, refer to the corresponding programming example.

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

Page Write Mode

Before a page write operation is executed, it is important to ensure that a relevant page erase operation has been successfully executed. The EEPROM page write operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page write. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM write enable control bit, namely WREN, is changed from “1” to “0”, the internal page buffer will also be cleared. Note that when the WREN bit is changed from “0” to “1”, the internal page buffer will not be cleared. A page write is initiated in the same way as a byte write initiation except that the EEPROM data can be written up to 16 bytes. The EEPROM address higher 6 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page write operation mode the lower 4-bit address value will automatically be incremented by one after each data byte is written into the EED register. However, the higher 6-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”. At this point any data write operations to the EED register will be invalid.

For page write operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers, then the data to be written should be placed in the EED register. The maximum data length for a page is 16 bytes. Note that when a data byte is written into the EED register, then the data in the EED register will be loaded into the internal page buffer and the current address value will automatically be incremented by one. When the page data is completely written into the page buffer, then the WREN bit in the EEC register should be set high to enable write operations and the WR bit must be immediately set high to initiate the EEPROM write process. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt enable bit EMI should also first be cleared before

implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

Note: The above steps must be executed sequentially to successfully complete the page write operation, refer to the corresponding programming example.

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

Write Protection

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

EEPROM Interrupt

The EEPROM erase or write interrupt is generated when an EEPROM erase or write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. However as the EEPROM interrupt is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM erase or write cycle ends, the DEF request flag and its associated multi-function interrupt request flag will be set. If the global and EEPROM 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, the multi-function interrupt flag will be automatically cleared, the EEPROM interrupt flag must be manually reset by the application program. The EMI bit will also be automatically cleared to disable other interrupts. More details can be obtained in the Interrupts section.

Programming Considerations

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

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

Programming Examples

Reading a Data Byte from the EEPROM - polling method

```
MOV A, 40H              ; setup memory pointer low byte MP1L
MOV MP1L, A             ; MP1 points to EEC register
MOV A, 01H              ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4              ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H   ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L   ; user defined low byte address
MOV EEAL, A
SET IAR1.1              ; set RDEN bit, enable read operations
SET IAR1.0              ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0              ; check for read cycle end
JMP BACK
CLR IAR1                ; disable EEPROM read function
CLR MP1H
MOV A, EED              ; move read data to register
MOV READ_DATA, A
```

Reading a Data Page from the EEPROM - polling method

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

Erasing a Data Page to the EEPROM − polling method

```
MOV A, 40H              ; setup memory pointer low byte MP1L
MOV MP1L, A             ; MP1 points to EEC register
MOV A, 01H              ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4              ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H   ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L   ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP Erase_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA      ; user defined data, erase mode don't care data value
MOV EED, A
RET
:
Erase_START:
CLR EMI
SET IAR1.6              ; set EREN bit, enable erase operations
SET IAR1.5              ; start Erase Cycle - set ER bit - executed immediately
                        ; after setting EREN bit
SET EMI
BACK:
SZ  IAR1.5              ; check for erase cycle end
JMP BACK
CLR MP1H
```

Writing a Data Byte to the EEPROM - polling method

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

Writing a Data Page to the EEPROM - polling method

```
MOV A, 40H              ; setup memory pointer low byte MP1L
MOV MP1L, A             ; MP1 points to EEC register
MOV A, 01H              ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4              ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H   ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L   ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP WRITE_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA      ; user defined data
MOV EED, A
RET
:
WRITE_START:
CLR EMI
SET IAR1.3              ; set WREN bit, enable write operations
SET IAR1.2              ; start Write Cycle - set WR bit - executed immediately
                        ; after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2              ; check for write cycle end
JMP BACK
CLR MP1H
```

Oscillators

Various oscillator options offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving. Oscillator selections and operation are selected through a combination of configuration option 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 oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency 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.

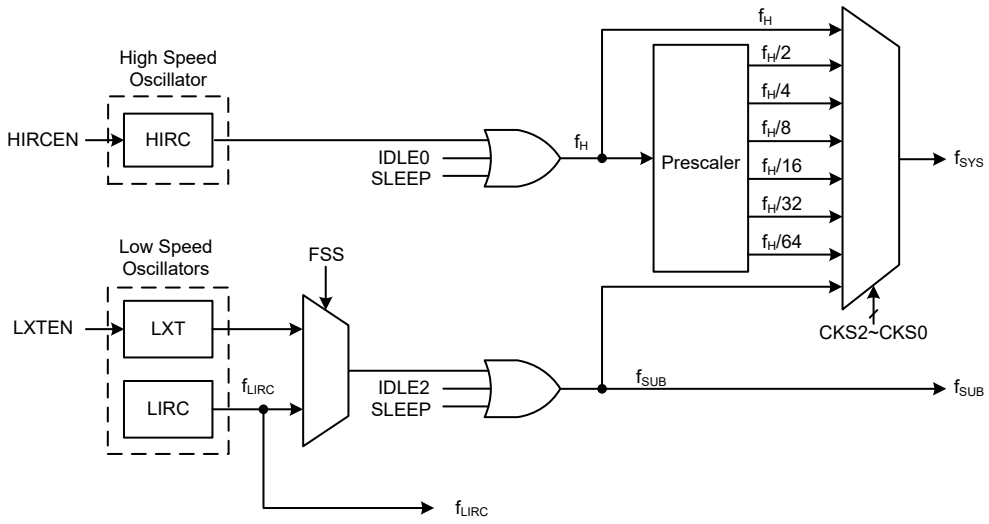
Type	Name	Frequency	Pins
Internal High Speed RC	HIRC	8/12/16MHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

System Clock Configurations

There are several oscillator sources, a high speed oscillators and two low speed oscillators. The high speed system clock is sourced from the internal 8/12/16MHz RC oscillator, HIRC. The low speed oscillators are the external 32.768kHz crystal oscillator, LXT, and the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

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



System Clock Configurations

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 8MHz, 12MHz and 16MHz, which are selected by HIRC1~HIRC0 bits in the HIRCC register. These bits 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 32.768 kHz 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.

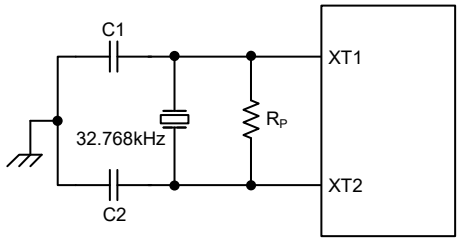
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE 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 Low Power Function

The LXT oscillator can function in one of three modes, the Low-Power Mode, Quick Start Mode and High ESR Mode. The mode selection is executed using the LXTSP1~LXTSP0 bits in the LXTC register.

LXTSP[1:0]		LXT Operating Mode
0	0	Low Power, C_L=7pF, ESR=30kΩ
0	1	Quick Start, C_L=7pF, ESR=30kΩ
1	0	High ESR, C_L=7pF, ESR=90kΩ (No Low Power/Quick Start distinction)
1	1	

When the LXTSP1~LXTSP0 bits are set to “01”, the LXT Quick Start Mode will be enabled. In the Quick Start Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up, it can be placed into the Low Power Mode by clearing the LXTSP1~LXTSP0 bits to zero. The oscillator will continue to run but with reduced current consumption. It is important to note that the LXT operating mode switching must be properly controlled before the LXT oscillator clock is selected as the system clock source. Once the LXT oscillator clock is selected as the system clock source using the CKS2~CKS0 bits and FSS bit in the SCC register, the LXT oscillator operating mode cannot be changed.

It should be note that no matter what condition the LXTSP1~LXTSP0 bits are set to, the LXT oscillator will be always function normally. The only difference is that it will take more time to start up if in the Low Power Mode.

Internal 32kHz Oscillator – LIRC

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

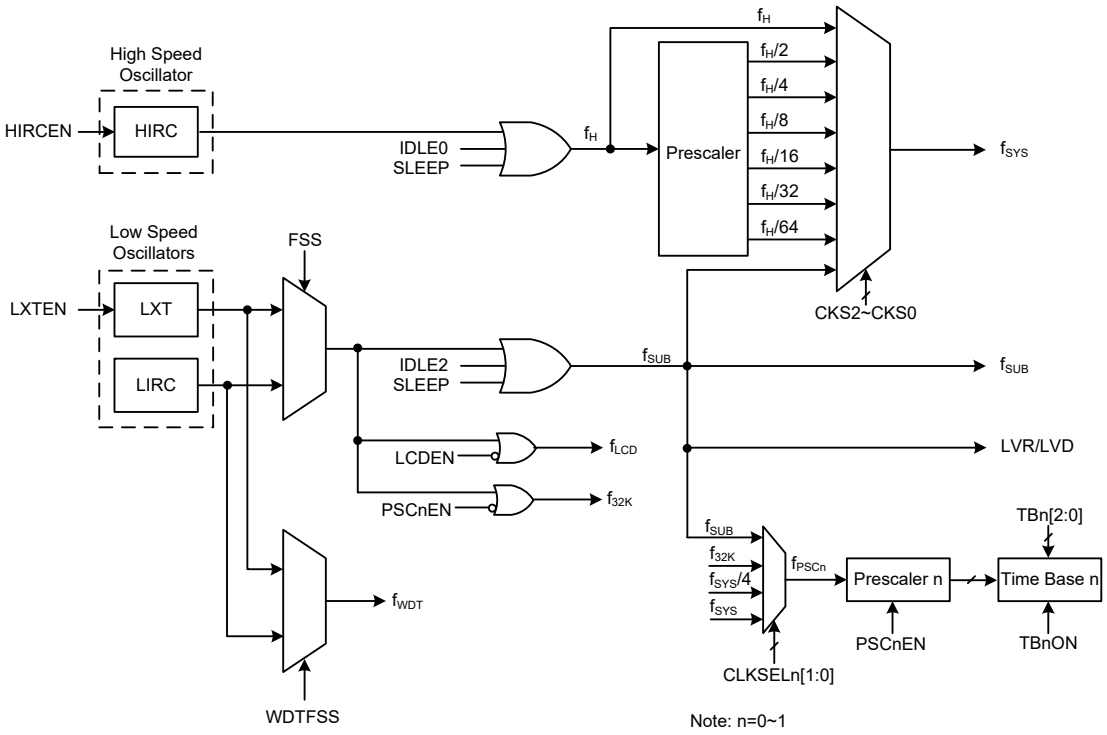
Operating Modes and System Clocks

Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice-versa lower speed clocks reduce current consumption. As 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 different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock selections using register programming, a clock system can be configured to obtain maximum application performance.

The main system clock can come from either a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source is 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: 1. When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillation can be stopped to conserve the power or continue to oscillate to provide the clock source, f_H~f_H/64, for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

2. The f_{WDT} clock on/off state is controlled by the WDT function being enabled or disabled using the WDTC register.
3. The f_{LCD} clock on/off state is controlled by the LCDEN bit in the LCDC0 register.
4. The f_{32K} clock on/off state is controlled by the PSCnEN bit in the PSCnR register.

System Operation Modes

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

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

"x": don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.
2. In the IDLE2 or SLEEP mode, if f_{LCD} on (from f_{LIRC}) or f_{32K} on (from f_{LIRC}) or f_{WDT} on (from f_{LIRC}), then f_{LIRC} is on; otherwise f_{LIRC} is always off.
3. In the IDLE2 or SLEEP mode, if f_{LCD} on (from f_{LXT}) or f_{32K} on (from f_{LXT}) or f_{WDT} on (from f_{LXT}), then f_{LXT} is on; otherwise f_{LXT} is always off (when LXTEN=1).

FAST Mode

This is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from the high speed oscillator, HIRC. 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 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_{LCD}, f_{32K} or f_{WDT} clock will continue to operate if the LCD, TimeBase or WDT function is enabled.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is low and the FSIDEN bit in the SCC register is high. In the IDLE0 Mode the CPU will be switched off but the low speed oscillator will be 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 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	WDTFSS	—	FSS	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
LXTC	—	—	—	—	LXTSP1	LXTSP0	LXTF	LXTEN

System Operating Mode Control Register List

• SCC Register

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

Bit 7~5 **CKS2~CKS0**: System clock selection
000: f_H
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: f_{SUB}

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

Bit 4 **WDTFSS**: Low frequency clock selection for f_{WDT}
0: LIRC
1: LXT

Bit 3 Unimplemented, read as "0"

Bit 2 **FSS**: Low frequency clock selection for f_{SUB}
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, WDTFSS or FSS bit. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.

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

• **HIRCC Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection
00: 8MHz
01: 12MHz
10: 16MHz
11: 8MHz

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 the 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

• **LXTC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	LXTSP1	LXTSP0	LXTF	LXTEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **LXTSP1~LXTPS0**: LXT mode control
00: Low Power mode
01: Quick Start mode
10/11: High ESR mode

These bits are used to control whether the LXT oscillator operates in the Low Power, Quick Start or High ESR mode. It is important to note that these bits cannot be changed after the LXT oscillator is selected as the system clock source and the LXTF bit is set high.

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

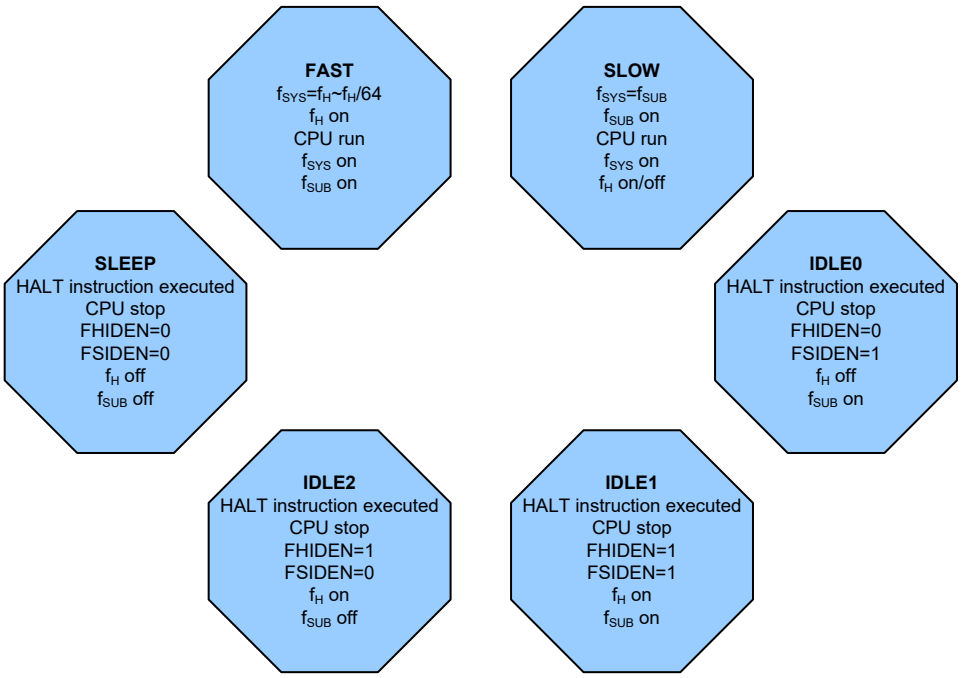
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set to 1 to enable the LXT oscillator, the LXTF bit will first be cleared to 0 and then set to 1 after the LXT oscillator is stable.

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

Operating Mode Switching

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

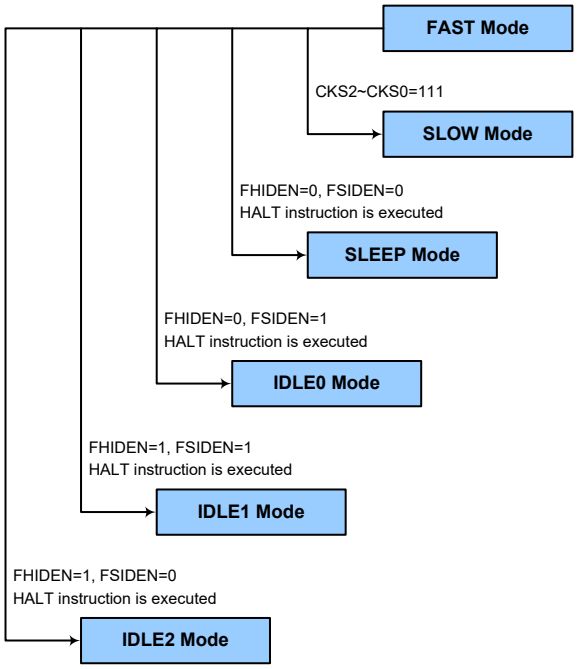
In simple terms, mode switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



FAST Mode to SLOW Mode Switching

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

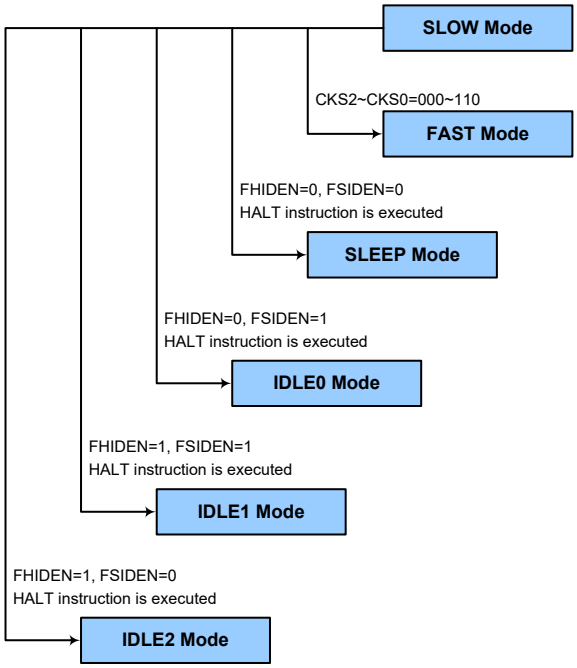
The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires the selected oscillator to be stable before full mode switching occurs.



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 $f_H \sim f_H/64$.

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



Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

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

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

Entering the IDLE2 Mode

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

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

Standby Current Considerations

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

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

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

Wake-up

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

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

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

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

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake up the system. When a Port A pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt

is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock, f_{WDT}, is provided by an internal clock which is sourced from either the LIRC or LXT oscillator, selected via the WDTFSS bit in the SCC register. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The LXT oscillator is supplied by an external 32.768 kHz crystal. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

The WDTC register controls the required time-out period as well as the WDT enable/disable and software 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 control
10101: Disable
01010: Enable
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set to 1.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^8/f_{WDT}$
001: $2^{10}/f_{WDT}$
010: $2^{12}/f_{WDT}$
011: $2^{14}/f_{WDT}$
100: $2^{15}/f_{WDT}$
101: $2^{16}/f_{WDT}$
110: $2^{17}/f_{WDT}$
111: $2^{18}/f_{WDT}$
These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period.

• **SCC Register**

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

Bit 7~5 **CKS2~CKS0**: System clock selection
Refer to the Operating Modes and System Clocks section.

Bit 4 **WDTFSS**: Low frequency clock selection for f_{WDT}
0: LIRC
1: LXT

Bit 3 Unimplemented, read as "0"

Bit 2 **FSS**: Low frequency clock selection for f_{SUB}
Refer to the Operating Modes and System Clocks section.

Bit 1 **FHIDEN**: High frequency oscillator control when CPU is switched off
Refer to the Operating Modes and System Clocks section.

Bit 0 **FSIDEN**: Low frequency oscillator control when CPU is switched off
Refer to the Operating Modes and System Clocks section.

• **RSTFC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	LVRF	LRF	WRF
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	x	0	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 **LRF**: LVR control register software reset flag
Refer to the Low Voltage Reset section.

Bit 0 **WRF**: WDT control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the WDT control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

Watchdog Timer Operation

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

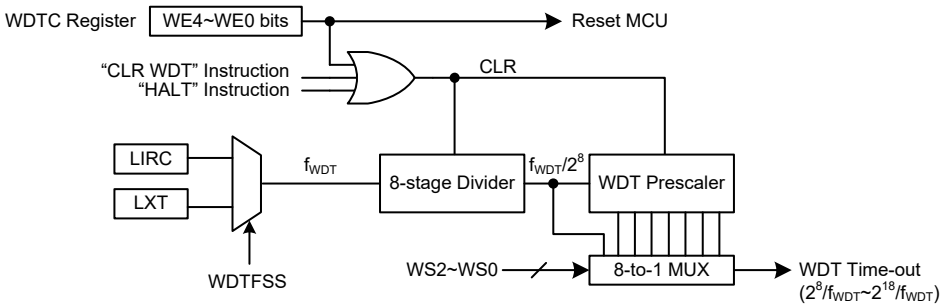
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

Watchdog Timer Function Control

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

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

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



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.

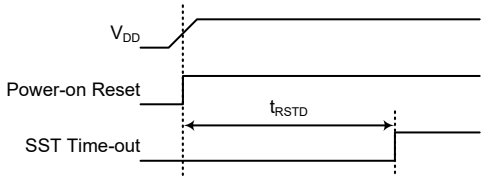
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

Reset Functions

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

Power-on Reset

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



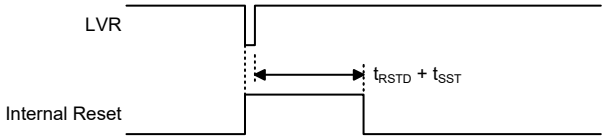
Power-On Reset Timing Chart

Low Voltage Reset – LVR

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

The LVR function can be enabled or disabled by the LVRC control register. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set 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 LVR&LVD Electrical Characteristics. If the duration of the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual t_{LVR} value can be selected by the TLVR1~TLVR0 bits in the TLVRC register. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits have any other values, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set to 1.

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



Low Voltage Reset Timing Chart

• **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage selection
01100110: 1.7V
01010101: 1.9V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
11110000: LVR disable
Other values: Generates an MCU reset – register is reset to POR value

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

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

• **TLVRC Register**

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

Bit 7~2 Unimplemented, read as "0"

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

• **RSTFC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	LVRF	LRF	WRF
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	x	0	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 high when a specific Low Voltage Reset situation occurs. This bit can only be cleared to zero by the application program.

Bit 1 **LRF**: LVR control register software reset flag
0: Not occurred
1: Occurred
This bit is set high if the LVRC register contains any non-defined LVRC register values. This in effect acts like a software-reset function. This bit can only be cleared to zero by the application program.

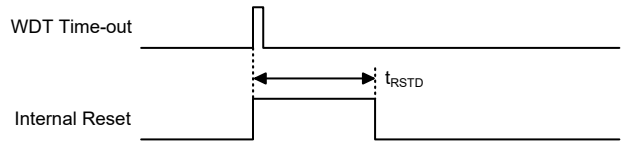
Bit 0 **WRF**: WDT control register software reset flag
Refer to the Watchdog Timer Control Register section.

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

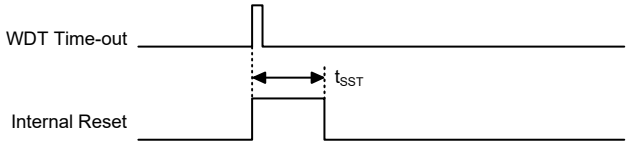
The Watchdog time-out Reset during normal operation in the FAST or SLOW Mode is the same as the hardware Low Voltage Reset except that the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

The Watchdog time-out Reset during SLEEP or IDLE Mode is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to "0" and the TO 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 Mode Timing Chart

Reset Initial Conditions

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

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

"u": unchanged

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

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

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

Register Name	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
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	---- -x00	---- -uuu	---- -uuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	0000 0000	0000 0000	uuuu uuuu
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
INTEG	---- 0000	---- 0000	---- uuuu
SCC	0000 -000	0000 -000	uuuu -uuu
HIRCC	---- 0001	---- 0001	---- uuuu
LXTC	---- 0000	---- 0000	---- uuuu
LVRC	0110 0110	0110 0110	uuuu uuuu
LVDC	0000 0000	0000 0000	uuuu uuuu
TLVRC	---- --01	---- --01	---- --uu
WDTC	0101 0011	0101 0011	uuuu uuuu

Register Name	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
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
MFI0	-000 -000	-000 -000	-uuu -uuu
MFI1	0000 0000	0000 0000	uuuu uuuu
MFI2	-000 -000	-000 -000	-uuu -uuu
MFI3	0000 0000	0000 0000	uuuu uuuu
MFI4	0000 0000	0000 0000	uuuu uuuu
PSC0R	---- -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	0000 0000	0000 0000	uuuu uuuu
SIMC2	0000 0000	0000 0000	uuuu uuuu
SIMTOC	0000 0000	0000 0000	uuuu uuuu
CTM0C0	0000 0000	0000 0000	uuuu uuuu
CTM0C1	0000 0000	0000 0000	uuuu uuuu
CTM0DL	0000 0000	0000 0000	uuuu uuuu
CTM0DH	---- --00	---- --00	---- --uu
CTM0AL	0000 0000	0000 0000	uuuu uuuu
CTM0AH	---- --00	---- --00	---- --uu
EEAL	0000 0000	0000 0000	uuuu uuuu
EEAH	---- --00	---- --00	---- --uu
EED	0000 0000	0000 0000	uuuu uuuu
PSC1R	---- -000	---- -000	---- -uuu
LCDC0	0000 0000	0000 0000	uuuu uuuu
LCDC1	000- 0000	000- 0000	uuu- uuuu
LCDC2	000- -000	000- -000	uuu- -uuu
PTM0C0	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	uuuu uuuu
PTM0DL	0000 0000	0000 0000	uuuu uuuu
PTM0DH	---- --00	---- --00	---- --uu
PTM0AL	0000 0000	0000 0000	uuuu uuuu
PTM0AH	---- --00	---- --00	---- --uu
PTM0RPL	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	---- --00	---- --00	---- --uu
FC0	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	uuuu uuuu
FC2	---- --00	---- --00	---- --uu
FARL	0000 0000	0000 0000	uuuu uuuu
FARH	--00 0000	--00 0000	--uu uuuu
FD0L	0000 0000	0000 0000	uuuu uuuu

Register Name	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
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
STMC0	0000 0---	0000 0---	uuuu u---
STMC1	0000 0000	0000 0000	uuuu uuuu
STMDL	0000 0000	0000 0000	uuuu uuuu
STMDH	0000 0000	0000 0000	uuuu uuuu
STMAL	0000 0000	0000 0000	uuuu uuuu
STMAH	0000 0000	0000 0000	uuuu uuuu
STMRP	0000 0000	0000 0000	uuuu uuuu
CTM1C0	0000 0000	0000 0000	uuuu uuuu
CTM1C1	0000 0000	0000 0000	uuuu uuuu
CTM1DL	0000 0000	0000 0000	uuuu uuuu
CTM1DH	---- --00	---- --00	---- --uu
CTM1AL	0000 0000	0000 0000	uuuu uuuu
CTM1AH	---- --00	---- --00	---- --uu
CTM2C0	0000 0000	0000 0000	uuuu uuuu
CTM2C1	0000 0000	0000 0000	uuuu uuuu
CTM2DL	0000 0000	0000 0000	uuuu uuuu
CTM2DH	---- --00	---- --00	---- --uu
CTM2AL	0000 0000	0000 0000	uuuu uuuu
CTM2AH	---- --00	---- --00	---- --uu
PE	1111 1111	1111 1111	uuuu uuuu
PEC	1111 1111	1111 1111	uuuu uuuu
PEPU	0000 0000	0000 0000	uuuu uuuu
PF	1111 1111	1111 1111	uuuu uuuu
PFC	1111 1111	1111 1111	uuuu uuuu
PFPU	0000 0000	0000 0000	uuuu uuuu
PG	1111 1111	1111 1111	uuuu uuuu
PGC	1111 1111	1111 1111	uuuu uuuu
PGPU	0000 0000	0000 0000	uuuu uuuu
PH	---- 1111	---- 1111	---- uuuu
PHC	---- 1111	---- 1111	---- uuuu
PHPU	---- 0000	---- 0000	---- uuuu
STKPTR	0--- 0000	0--- 0000	u--- 0000
PMPS	--00 0000	--00 0000	--uu uuuu
ORMC	0000 0000	0000 0000	0000 0000
TKTMR	0000 0000	0000 0000	uuuu uuuu
TKC0	0000 0-00	0000 0-00	uuuu u-uu
TK16DL	0000 0000	0000 0000	uuuu uuuu
TK16DH	0000 0000	0000 0000	uuuu uuuu
TKC1	0000 0011	0000 0011	uuuu uuuu
TKM016DL	0000 0000	0000 0000	uuuu uuuu

Register Name	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TKM016DH	0000 0000	0000 0000	uuuu uuuu
TKM0ROL	0000 0000	0000 0000	uuuu uuuu
TKM0ROH	---- --00	---- --00	---- --uu
TKM0C0	--00 0000	--00 0000	--uu uuuu
TKM0C1	0-00 0000	0-00 0000	u-uu uuuu
TKM0C2	1110 0100	1110 0100	uuuu uuuu
TKM116DL	0000 0000	0000 0000	uuuu uuuu
TKM116DH	0000 0000	0000 0000	uuuu uuuu
TKM1ROL	0000 0000	0000 0000	uuuu uuuu
TKM1ROH	---- --00	---- --00	---- --uu
TKM1C0	--00 0000	--00 0000	--uu uuuu
TKM1C1	0-00 0000	0-00 0000	u-uu uuuu
TKM1C2	1110 0100	1110 0100	uuuu uuuu
TKM216DL	0000 0000	0000 0000	uuuu uuuu
TKM216DH	0000 0000	0000 0000	uuuu uuuu
TKM2ROL	0000 0000	0000 0000	uuuu uuuu
TKM2ROH	---- --00	---- --00	---- --uu
TKM2C0	--00 0000	--00 0000	--uu uuuu
TKM2C1	0-00 0000	0-00 0000	u-uu uuuu
TKM2C2	1110 0100	1110 0100	uuuu uuuu
TKM316DL	0000 0000	0000 0000	uuuu uuuu
TKM316DH	0000 0000	0000 0000	uuuu uuuu
TKM3ROL	0000 0000	0000 0000	uuuu uuuu
TKM3ROH	---- --00	---- --00	---- --uu
TKM3C0	--00 0000	--00 0000	--uu uuuu
TKM3C1	0-00 0000	0-00 0000	u-uu uuuu
TKM3C2	1110 0100	1110 0100	uuuu uuuu
TKM416DL	0000 0000	0000 0000	uuuu uuuu
TKM416DH	0000 0000	0000 0000	uuuu uuuu
TKM4ROL	0000 0000	0000 0000	uuuu uuuu
TKM4ROH	---- --00	---- --00	---- --uu
TKM4C0	--00 0000	--00 0000	--uu uuuu
TKM4C1	0-00 0000	0-00 0000	u-uu uuuu
TKM4C2	1110 0100	1110 0100	uuuu uuuu
TKM516DL	0000 0000	0000 0000	uuuu uuuu
TKM516DH	0000 0000	0000 0000	uuuu uuuu
TKM5ROL	0000 0000	0000 0000	uuuu uuuu
TKM5ROH	---- --00	---- --00	---- --uu
TKM5C0	--00 0000	--00 0000	--uu uuuu
TKM5C1	0-00 0000	0-00 0000	u-uu uuuu
TKM5C2	1110 0100	1110 0100	uuuu uuuu
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

Register Name	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PTM1RPL	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	---- --00	---- --00	---- --uu
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
EEC	0000 0000	0000 0000	uuuu uuuu
IFS0	--00 0000	--00 0000	--uu uuuu
IFS1	0000 0000	0000 0000	uuuu uuuu
IFS2	0000 0000	0000 0000	uuuu uuuu
IFS3	0000 0000	0000 0000	uuuu uuuu
PAS0	-000 -000	-000 -000	-uuu -uuu
PAS1	-000 -000	-000 -000	-uuu -uuu
PAS2	-000 -000	-000 -000	-uuu -uuu
PAS3	-000 -000	-000 -000	-uuu -uuu
PBS0	-000 -000	-000 -000	-uuu -uuu
PBS1	-000 -000	-000 -000	-uuu -uuu
PBS2	-000 -000	-000 -000	-uuu -uuu
PBS3	-000 -000	-000 -000	-uuu -uuu
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
PES0	-000 -000	-000 -000	-uuu -uuu
PES1	-000 -000	-000 -000	-uuu -uuu
PES2	-000 -000	-000 -000	-uuu -uuu
PES3	-000 -000	-000 -000	-uuu -uuu
PFS0	-000 -000	-000 -000	-uuu -uuu
PFS1	-000 -000	-000 -000	-uuu -uuu
PFS2	-000 -000	-000 -000	-uuu -uuu
PFS3	-000 -000	-000 -000	-uuu -uuu
PGS0	0000 0000	0000 0000	uuuu uuuu
PGS1	0000 0000	0000 0000	uuuu uuuu
PHS0	0000 0000	0000 0000	uuuu uuuu
IFS4	---0 0000	---0 0000	---u uuuu
PCRL	0000 0000	0000 0000	uuuu uuuu
PCRH	--00 0000	--00 0000	--uu uuuu
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 -000	0000 -000	uuuu -uuu
SADC2	00-0 0000	00-0 0000	uu-u uuuu
SADC3	---- --00	---- --00	---- --xx
SADC4	00-0 -000	00-0 -000	uu-u -uuu

Register Name	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
SADO1BL	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADO1BH	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADO2BL	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADO2BH	xxxx xxxx	xxxx xxxx	uuuu uuuu
LEBC	0000 0000	0000 0000	uuuu uuuu
ATAC1C	--00 0000	--00 0000	--uu uuuu
ATAC2C	---0 0000	---0 0000	---u uuuu
ATADT	---- 0000	---- 0000	---- uuuu
TCRL	0000 0000	0000 0000	uuuu uuuu
TCRH	0000 0000	0000 0000	uuuu uuuu
TCRC	---0 --00	---0 --00	---u --uu
SLEDC0	--01 0010	--01 0010	--uu uuuu
SLEDC1	--01 0010	--01 0010	--uu uuuu
SLEDC2	--01 0010	--01 0010	--uu uuuu
SLEDC3	--01 0010	--01 0010	--uu uuuu
SLEDC4	--01 0010	--01 0010	--uu uuuu
SLEDC5	--01 0010	--01 0010	--uu uuuu
SLEDC6	--01 0010	--01 0010	--uu uuuu
SLEDC7	---- -010	---- -010	---- -uuu
LVPUC	---- ---0	---- ---0	---- ---u
IECC	0000 0000	0000 0000	uuuu uuuu
OVPC0	000- -000	000- -000	uuu- -uuu
OVPC1	0001 0000	0001 0000	uuuu uuuu
OVPC2	0--- 0000	0--- 0000	u--- uuuu
OVPDA	0000 0000	0000 0000	uuuu uuuu
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
RxCNT1	---- -000	---- -000	---- -uuu
U2SR	0000 1011	0000 1011	uuuu uuuu
U2CR1	0000 00x0	0000 00x0	uuuu uuuu
U2CR2	0000 0000	0000 0000	uuuu uuuu
U2CR3	---- ---0	---- ---0	---- ---u
BRDH2	0000 0000	0000 0000	uuuu uuuu
BRDL2	0000 0000	0000 0000	uuuu uuuu

Register Name	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
UFCR2	--00 0000	--00 0000	--uu uuuu
TXR_RXR2	xxxx xxxx	xxxx xxxx	uuuu uuuu
RxCNT2	---- -000	---- -000	---- -uuu
CPR	0000 0000	0000 0000	uuuu uuuu
OCVPC	---- 1000	---- 1000	---- uuuu
CRCCR	---- 0000	---- 0000	---- uuuu
CRCIN	0000 0000	0000 0000	uuuu uuuu
CRCDL	0000 0000	0000 0000	uuuu uuuu
CRCDH	0000 0000	0000 0000	uuuu uuuu
IICC0	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	0000 000-	0000 000-	uuuu uuu-
IICTOC	0000 0000	0000 0000	uuuu uuuu
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

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~PH. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory Structure diagram. The I/O port can be used for input and output operations. For input operation, the port is 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
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PF	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0
PG	PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0
PGC	PGC7	PGC6	PGC5	PGC4	PGC3	PGC2	PGC1	PGC0
PGPU	PGPU7	PGPU6	PGPU5	PGPU4	PGPU3	PGPU2	PGPU1	PGPU0
PH	—	—	—	—	PH3	PH2	PH1	PH0
PHC	—	—	—	—	PHC3	PHC2	PHC1	PHC0
PHPU	—	—	—	—	PHPU3	PHPU2	PHPU1	PHPU0

"—": 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 the PxPU and LVPUC registers, and are implemented using weak PMOS transistors. The PxPU register is used to determine whether the pull-high function is enabled or not while the LVPUC register is used to select the pull-high resistor value for low voltage power supply applications.

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

• **PxPU Register**

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

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

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

• **LVPUC Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **LVPU**: Pull-high resistor selection for low voltage power supply
0: All pin pull-high resistors are 60kΩ @ 3V
1: All pin pull-high resistors are 15kΩ @ 3V

The LVPUC register is used to select the pull-high resistor value for low voltage power supply applications. Note that the LVPU bit is only available when the corresponding pin pull-high function is enabled. This bit will have no effect on selecting the pull-high resistor value when the pull-high function is disabled.

Port A Wake-Up

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

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

• **PAWU Register**

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

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

I/O Port Control Register

Each I/O Port has its own control register known as PAC~PHC, 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 cleared to "0".

• PxC Register

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

PxCn: I/O Port x Pin type selection

0: Output
1: Input

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

I/O Port Source Current Selection

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLEDC0	—	—	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
SLEDC1	—	—	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
SLEDC2	—	—	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20
SLEDC3	—	—	SLEDC35	SLEDC34	SLEDC33	SLEDC32	SLEDC31	SLEDC30
SLEDC4	—	—	SLEDC45	SLEDC44	SLEDC43	SLEDC42	SLEDC41	SLEDC40
SLEDC5	—	—	SLEDC55	SLEDC54	SLEDC53	SLEDC52	SLEDC51	SLEDC50
SLEDC6	—	—	SLEDC65	SLEDC64	SLEDC63	SLEDC62	SLEDC61	SLEDC60
SLEDC7	—	—	—	—	—	SLEDC72	SLEDC71	SLEDC70

I/O Port Source Current Selection Register List

• SLEDC0 Register

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **SLEDC05~SLEDC03**: PA7~PA4 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

Bit 2~0 **SLEDC02~SLEDC00**: PA3~PA0 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

• **SLEDC1 Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **SLEDC15~SLEDC13**: PB7~PB4 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

Bit 2~0 **SLEDC12~SLEDC10**: PB3~PB0 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

SLEDC2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	1	0	0	1	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **SLEDC25~SLEDC23**: PC7~PC4 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2

011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

Bit 2~0 **SLEDC22~SLEDC20**: PC3~PC0 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

• SLEDC3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	SLEDC35	SLEDC34	SLEDC33	SLEDC32	SLEDC31	SLEDC30
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	1	0	0	1	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **SLEDC35~SLEDC33**: PD6~PD4 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

Bit 2~0 **SLEDC32~SLEDC30**: PD3~PD0 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

• SLEDC4 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	SLEDC45	SLEDC44	SLEDC43	SLEDC42	SLEDC41	SLEDC40
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	1	0	0	1	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **SLEDC45~SLEDC43**: PE7~PE4 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

Bit 2~0 **SLEDC42~SLEDC40**: PE3~PE0 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

• **SLEDC5 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SLEDC55	SLEDC54	SLEDC53	SLEDC52	SLEDC51	SLEDC50
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	1	0	0	1	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **SLEDC55~SLEDC53**: PF7~PF4 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

Bit 2~0 **SLEDC52~SLEDC50**: PF3~PF0 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

• **SLEDC6 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SLEDC65	SLEDC64	SLEDC63	SLEDC62	SLEDC61	SLEDC60
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	1	0	0	1	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **SLEDC65~SLEDC63**: PG7~PG4 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

Bit 2~0 **SLEDC62~SLEDC60**: PG3~PG0 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2

011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

• **SLEDC7 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	SLEDC72	SLEDC71	SLEDC70
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	1	0

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **SLEDC72~SLEDC70**: PH3~PH0 source current selection
000: source current=Level 0 (Min.)
001: source current=Level 1
010: source current=Level 2
011: source current=Level 3
100: source current=Level 4
101: source current=Level 5
110: source current=Level 6
111: source current=Level 7 (Max.)

I/O Port Power Source Control

This device supports different I/O port power source selections for PB1~PB3. 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 either the power pin VDD or VDDIO, which is determined using the PMPS5~PMPS0 bits in the PMPS register. The VDDIO power pin function should first be selected using the corresponding pin-shared function selection bits if the port power is supposed to come from the VDDIO pin.

An important point to know is that the input power voltage on the VDDIO 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. When the V_{DDIO} is less than 2.2V, it is recommended that the V_{DD} is equal to the V_{DDIO}. Because when the V_{DDIO} is less than 2.2V, the I_{OH} thrust is too low if the V_{DD} is larger than the V_{DDIO}.

• **PMPS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PMPS5	PMPS4	PMPS3	PMPS2	PMPS1	PMPS0
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 **PMPS5~PMPS4**: PB3 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}
If the PB0 pin is switched to the VDDIO function, and the PMPS5~PMPS4 bits are set to "1x", the VDDIO pin input voltage can be used for PB3 pin power.

Bit 3~2 **PMPS3~PMPS2**: PB2 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}
If the PB0 pin is switched to the VDDIO function, and the PMPS3~PMPS2 bits are set to "1x", the VDDIO pin input voltage can be used for PB2 pin power.

Bit 1~0 **PMPS1~PMPS0**: PB1 pin power supply selection
0x: V_{DD}
1x: V_{DDIO}
If the PB0 pin is switched to the VDDIO function, and the PMPS1~PMPS0 bits are set to "1x", the VDDIO pin input voltage can be used for PB1 pin power.

Pin-shared Functions

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

Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The device includes Port "x" output function Selection register "n", labeled as 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 setup as input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	—	PAS06	PAS05	PAS04	—	PAS02	PAS01	PAS00
PAS1	—	PAS16	PAS15	PAS14	—	PAS12	PAS11	PAS10
PAS2	—	PAS26	PAS25	PAS24	—	PAS22	PAS21	PAS20
PAS3	—	PAS36	PAS35	PAS34	—	PAS32	PAS31	PAS30
PBS0	—	PBS06	PBS05	PBS04	—	PBS02	PBS01	PBS00
PBS1	—	PBS16	PBS15	PBS14	—	PBS12	PBS11	PBS10
PBS2	—	PBS26	PBS25	PBS24	—	PBS22	PBS21	PBS20
PBS3	—	PBS36	PBS35	PBS34	—	PBS32	PBS31	PBS30
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
PES0	—	PES06	PES05	PES04	—	PES02	PES01	PES00
PES1	—	PES16	PES15	PES14	—	PES12	PES11	PES10
PES2	—	PES26	PES25	PES24	—	PES22	PES21	PES20
PES3	—	PES36	PES35	PES34	—	PES32	PES31	PES30

Register Name	Bit							
	7	6	5	4	3	2	1	0
PFS0	—	PFS06	PFS05	PFS04	—	PFS02	PFS01	PFS00
PFS1	—	PFS16	PFS15	PFS14	—	PFS12	PFS11	PFS10
PFS2	—	PFS26	PFS25	PFS24	—	PFS22	PFS21	PFS20
PFS3	—	PFS36	PFS35	PFS34	—	PFS32	PFS31	PFS30
PGS0	PGS07	PGS06	PGS05	PGS04	PGS03	PGS02	PGS01	PGS00
PGS1	PGS17	PGS16	PGS15	PGS14	PGS13	PGS12	PGS11	PGS10
PHS0	PHS07	PHS06	PHS05	PHS04	PHS03	PHS02	PHS01	PHS00
IFS0	—	—	SDISDA2PS	SDISDA1PS	SDISDA0PS	SCKSCL2PS	SCKSCL1PS	SCKSCL0PS
IFS1	INT01PS	INT00PS	RX12PS	RX11PS	RX10PS	RX02PS	RX01PS	RX00PS
IFS2	PTP0IPS	PTP2IPS	STPIPS	CTCK01PS	CTCK00PS	RX22PS	RX21PS	RX20PS
IFS3	IICSDA3PS	IICSDA2PS	IICSDA1PS	IICSDA0PS	IICSCL3PS	IICSCL2PS	IICSCL1PS	IICSCL0PS
IFS4	—	—	—	INT11PS	INT10PS	SCSB2PS	SCSB1PS	SCSB0PS

Pin-shared Function Selection Register List

- **PAS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PAS06	PAS05	PAS04	—	PAS02	PAS01	PAS00
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 6~4 **PAS06~PAS04**: PA1 pin-shared function selection
000: PA1
001: CTP0
010: V2
011: SEG42
100: IICSDA
101: PA1
110: PA1
111: PA1

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PAS02~PAS00**: PA0 pin-shared function selection
000: PA0
001: SDO
010: CTP0B
011: TX1
100: IICSCL
101: PA0
110: PA0
111: PA0

- **PAS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PAS16	PAS15	PAS14	—	PAS12	PAS11	PAS10
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 6~4 **PAS16~PAS14**: PA3 pin-shared function selection
000: PA3
001: $\overline{\text{SCS}}$
010: XT1

011: RX0/TX0
100: PA3
101: PA3
110: PA3
111: PA3

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PAS12~PAS10**: PA2 pin-shared function selection
000: PA2
001: $\overline{\text{SCS}}$
010: RX1/TX1
011: IICSDA
100: PA2
101: PA2
110: PA2
111: PA2

• **PAS2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PAS26	PAS25	PAS24	—	PAS22	PAS21	PAS20
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 6~4 **PAS26~PAS24**: PA5 pin-shared function selection
000: PA5
001: SEG15
010: KEY20
011: RX2/TX2
100: PA5
101: PA5
110: PA5
111: PA5

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PAS22~PAS20**: PA4 pin-shared function selection
000: PA4
001: SDO
010: XT2
011: TX0
100: PA4
101: PA4
110: PA4
111: PA4

• **PAS3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PAS36	PAS35	PAS34	—	PAS32	PAS31	PAS30
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 6~4 **PAS36~PAS34**: PA7 pin-shared function selection
000: PA7
001: CTP0B
010: TX1
011: C1
100: SEG41

101: IICSCL
110: PA7
111: PA7

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PAS32~PAS30**: PA6 pin-shared function selection
000: PA6/CTCK0/INT0
001: RX1/TX1
010: C2
011: SEG40
100: PA6/CTCK0/INT0
101: PA6/CTCK0/INT0
110: PA6/CTCK0/INT0
111: PA6/CTCK0/INT0

• **PBS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PBS06	PBS05	PBS04	—	PBS02	PBS01	PBS00
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 6~4 **PBS06~PBS04**: PB1 pin-shared function selection
000: PB1/INT0
001: SCK/SCL
010: VREFI
011: AN1
100: IICSCL
101: RX1/TX1
110: PB1/INT0
111: PB1/INT0

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PBS02~PBS00**: PB0 pin-shared function selection
000: PB0
001: SDI/SDA
010: VREF
011: AN0
100: IICSDA
101: TX1
110: VDDIO
111: PB0

• **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PBS16	PBS15	PBS14	—	PBS12	PBS11	PBS10
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 6~4 **PBS16~PBS14**: PB3 pin-shared function selection
000: PB3/INT1
001: $\overline{SCS}$
010: RX2/TX2
011: AN3
100: IICSCL
101: OVPINP

110: OVPCOUT
111: OVPINT

Note that it is necessary to avoid setting PB3 and PB6 to OVPINP input at the same time.

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PBS12~PBS10**: PB2 pin-shared function selection
000: PB2/PTP0I
001: SDO
010: TX2
011: AN2
100: IICSDA
101: OVPINN
110: OVPCOUT
111: PTP0

Note that it is necessary to avoid setting PB2 and PB5 to OVPINN input at the same time.

• PBS2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	PBS26	PBS25	PBS24	—	PBS22	PBS21	PBS20
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 6~4 **PBS26~PBS24**: PB5 pin-shared function selection
000: PB5/STCK
001: OVPINN
010: KEY2
011: AN5
100: PB5/STCK
101: PB5/STCK
110: PB5/STCK
111: PB5/STCK

Note that it is necessary to avoid setting PB2 and PB5 to OVPINN input at the same time.

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PBS22~PBS20**: PB4 pin-shared function selection
000: PB4/PTP0I
001: PTP0B
010: KEY1
011: AN4
100: OVPCOUT
101: OVPINT
110: PB4/PTP0I
111: PB4/PTP0I

• PBS3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	PBS36	PBS35	PBS34	—	PBS32	PBS31	PBS30
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 6~4 **PBS36~PBS34**: PB7 pin-shared function selection
000: PB7/INT1
001: PB7/INT1

010: KEY4
011: AN7
100: PB7/INT1
101: PB7/INT1
110: PB7/INT1
111: PB7/INT1

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PBS32~PBS30**: PB6 pin-shared function selection
000: PB6/PTCK0
001: OVPINP
010: KEY3
011: AN6
100: PB6/PTCK0
101: PB6/PTCK0
110: PB6/PTCK0
111: PB6/PTCK0

Note that it is necessary to avoid setting PB3 and PB6 to OVPINP input at the same time.

• 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: SEG3
10: KEY8
11: PC3

Bit 5~4 **PCS05~PCS04**: PC2 pin-shared function selection
00: PC2
01: SEG2
10: KEY7
11: PC2

Bit 3~2 **PCS03~PCS02**: PC1 pin-shared function selection
00: PC1
01: SEG1
10: KEY6
11: PC1

Bit 1~0 **PCS01~PCS00**: PC0 pin-shared function selection
00: PC0
01: SEG0
10: KEY5
11: PC0

• 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: SEG7

10: KEY12
11: IICSCL

Bit 5~4 **PCS15~PCS14**: PC6 pin-shared function selection
00: PC6
01: SEG6
10: KEY11
11: IICSDA

Bit 3~2 **PCS13~PCS12**: PC5 pin-shared function selection
00: PC5
01: SEG5
10: KEY10
11: PWML

Bit 1~0 **PCS11~PCS10**: PC4 pin-shared function selection
00: PC4
01: SEG4
10: KEY9
11: PWMH

• **PDS0 Register**

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

Bit 7~6 **PDS07~PDS06**: PD3 pin-shared function selection
00: PD3
01: SEG11
10: KEY16
11: PD3

Bit 5~4 **PDS05~PDS04**: PD2 pin-shared function selection
00: PD2
01: SEG10
10: KEY15
11: PD2

Bit 3~2 **PDS03~PDS02**: PD1 pin-shared function selection
00: PD1
01: SEG9
10: KEY14
11: PD1

Bit 1~0 **PDS01~PDS00**: PD0 pin-shared function selection
00: PD0
01: SEG8
10: KEY13
11: PD0

• **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: SEG14

10: KEY19
11: TX2

Bit 3~2 **PDS13~PDS12**: PD5 pin-shared function selection
00: PD5
01: SEG13
10: KEY18
11: TX1

Bit 1~0 **PDS11~PDS10**: PD4 pin-shared function selection
00: PD4
01: SEG12
10: KEY17
11: RX1/TX1

• **PES0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PES06	PES05	PES04	—	PES02	PES01	PES00
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 5~4 **PES06~PES04**: PE1 pin-shared function selection
000: PE1/PTP2I
001: SEG17
010: KEY22
011: PTP2
100: PWML
101: PE1/PTP2I
110: PE1/PTP2I
111: PE1/PTP2I

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PES02~PES00**: PE0 pin-shared function selection
000: PE0
001: SEG16
010: KEY21
011: PTP2B
100: PWMH
101: PE0
110: PE0
111: PE0

• **PES1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PES16	PES15	PES14	—	PES12	PES11	PES10
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 5~4 **PES16~PES14**: PE3 pin-shared function selection
000: PE3/STPI
001: SEG19
010: KEY24
011: $\overline{\text{RX0}}$/TX0
100: $\overline{\text{SCS}}$
101: STPB
110: PE3/STPI
111: PE3/STPI

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PES12~PES10**: PE2 pin-shared function selection
000: PE2/STPI/PTCK2
001: SEG18
010: KEY23
011: TX0
100: STP
101: PE2/STPI/PTCK2
110: PE2/STPI/PTCK2
111: PE2/STPI/PTCK2

• **PES2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PES26	PES25	PES24	—	PES22	PES21	PES20
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 5~4 **PES26~PES24**: PE5 pin-shared function selection
000: PE5/CTCK1
001: SEG21
010: RX2/TX2
011: IICSCL
100: SDI/SDA
101: CTP0
110: PE5/CTCK1
111: PE5/CTCK1

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PES22~PES20**: PE4 pin-shared function selection
000: PE4/PTP2I
001: SEG20
010: TX2
011: IICSDA
100: SCK/SCL
101: SDO
110: PTP0
111: PE4/PTP2I

• **PES3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PES36	PES35	PES34	—	PES32	PES31	PES30
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 5~4 **PES36~PES34**: PE7 pin-shared function selection
000: PE7/CTCK0/INT1
001: SEG23
010: CTP1B
011: $\overline{\text{RX1}}$/TX1
100: $\overline{\text{SCS}}$
101: STP
110: PE7/CTCK0/INT1
111: PE7/CTCK0/INT1

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PES32~PES30**: PE6 pin-shared function selection
000: PE6
001: SEG22
010: CTP1
011: TX1
100: SCK/SCL
101: SDO
110: PE6
111: PE6

• **PFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PFS06	PFS05	PFS04	—	PFS02	PFS01	PFS00
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 5~4 **PFS06~PFS04**: PF1 pin-shared function selection
000: PF1
001: SEG25
010: CTP0
011: RX0/TX0
100: SDI/SDA
101: PF1
110: IICSCL
111: PF1
Bit 3 Unimplemented, read as "0"
Bit 2~0 **PFS02~PFS00**: PF0 pin-shared function selection
000: PF0
001: SEG24
010: CTP0B
011: TX0
100: SDO
101: SCK/SCL
110: IICSDA
111: PF0

• **PFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PFS16	PFS15	PFS14	—	PFS12	PFS11	PFS10
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 5~4 **PFS16~PFS14**: PF3 pin-shared function selection
000: PF3/CTCK0/INT0
001: SEG27
010: RX1/TX1
011: $\overline{\text{SCS}}$
100: PTP2
101: PF3/CTCK0/INT0
110: PF3/CTCK0/INT0
111: PF3/CTCK0/INT0
Bit 3 Unimplemented, read as "0"

Bit 2~0 **PFS12~PFS10**: PF2 pin-shared function selection
000: PF2
001: SEG26
010: TX1
011: SCK/SCL
100: SDO
101: CTP0B
110: PF2
111: PF2

• **PFS2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PFS26	PFS25	PFS24	—	PFS22	PFS21	PFS20
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 5~4 **PFS26~PFS24**: PF5 pin-shared function selection
000: PF5
001: SEG29
010: CTP2
011: SDI/SDA
100: PTP1
101: PTP2
110: PF5
111: PF5
Bit 3 Unimplemented, read as "0"
Bit 2~0 **PFS22~PFS20**: PF4 pin-shared function selection
000: PF4/CTCK2
001: SEG28
010: PTP1B
011: SCK/SCL
100: PF4/CTCK2
101: PF4/CTCK2
110: PF4/CTCK2
111: PF4/CTCK2

• **PFS3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PFS36	PFS35	PFS34	—	PFS32	PFS31	PFS30
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 5~4 **PFS36~PFS34**: PF7 pin-shared function selection
000: PF7/PTP1I
001: SEG31
010: $\overline{\text{RX2}}$/TX2
011: $\overline{\text{SCS}}$
100: PWML
101: IICSCL
110: IICSDA
111: PF7/PTP1I
Bit 3 Unimplemented, read as "0"

Bit 2~0 **PFS32~PFS30**: PF6 pin-shared function selection
000: PF6/PTCK1
001: SEG30
010: TX2
011: SDO
100: PWMH
101: IICSDA
110: IICSCL
111: CTP2B

• **PGS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PGS07	PGS06	PGS05	PGS04	PGS03	PGS02	PGS01	PGS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PGS07~PGS06**: PG3 pin-shared function selection
00: PG3
01: SEG35
10: COM4
11: IICSCL

Bit 5~4 **PGS05~PGS04**: PG2 pin-shared function selection
00: PG2
01: SEG34
10: COM5
11: IICSDA

Bit 3~2 **PGS03~PGS02**: PG1 pin-shared function selection
00: PG1
01: SEG33
10: COM6
11: TX2

Bit 1~0 **PGS01~PGS00**: PG0 pin-shared function selection
00: PG0
01: SEG32
10: COM7
11: RX2/TX2

• **PGS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PGS17	PGS16	PGS15	PGS14	PGS13	PGS12	PGS11	PGS10
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 **PGS17~PGS16**: PG7 pin-shared function selection
00: PG7
01: SEG39
10: TX0
11: SCK/SCL

Bit 5~4 **PGS15~PGS14**: PG6 pin-shared function selection
00: PG6
01: SEG38
10: RX0/TX0
11: SDI/SDA

Bit 3~2 **PGS13~PGS12**: PG5 pin-shared function selection
00: PG5
01: SEG37

10: RX1/TX1
11: SDO

Bit 1~0 **PGS11~PGS10**: PG4 pin-shared function selection
00: PG4
01: SEG36
10: TX1
11: $\overline{SCS}$

• **PHS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PHS07	PHS06	PHS05	PHS04	PHS03	PHS02	PHS01	PHS00
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 **PHS07~PHS06**: PH3 pin-shared function selection
00: PH3
01: COM0
10: PH3
11: PH3

Bit 5~4 **PHS05~PHS04**: PH2 pin-shared function selection
00: PH2
01: COM1
10: PH2
11: PH2

Bit 3~2 **PHS03~PHS02**: PH1 pin-shared function selection
00: PH1
01: COM2
10: PH1
11: PH1

Bit 1~0 **PHS01~PHS00**: PH0 pin-shared function selection
00: PH0
01: COM3
10: PH0
11: PH0

• **IFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SDISDA2PS	SDISDA1PS	SDISDA0PS	SCKSCL2PS	SCKSCL1PS	SCKSCL0PS
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 **SDISDA2PS~SDISDA0PS**: SDI/SDA input source pin selection
000: PB0
001: PE5
010: PF1
011: PF5
100: PG6
101: PB0
110: PB0
111: PB0

Bit 2~0 **SCKSCL2PS~SCKSCL0PS**: SCK/SCL input source pin selection
000: PB1
001: PE4
010: PE6
011: PF0

100: PF2
101: PF4
110: PG7
111: PB1

• **IFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	INT01PS	INT00PS	RX12PS	RX11PS	RX10PS	RX02PS	RX01PS	RX00PS
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 **INT01PS~INT00PS**: INT0 input source pin selection
00: PA6
01: PF3
10: PB1
11: PB1

Bit 5~3 **RX12PS~RX10PS**: RX1/TX1 input source pin selection
000: PA6
001: PB1
010: PD4
011: PE7
100: PF3
101: PG5
110: PA2
111: PA6

Bit 2~0 **RX02PS~RX00PS**: RX0/TX0 input source pin selection
000: PG6
001: PA3
010: PE3
011: PF1
100: PG6
101: PG6
110: PG6
111: PG6

• **IFS2 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTP0IPS	PTP2IPS	STPIPS	CTCK01PS	CTCK00PS	RX22PS	RX21PS	RX20PS
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 **PTP0IPS**: PTP0I input source pin selection
0: PB2
1: PB4

Bit 6 **PTP2IPS**: PTP2I input source pin selection
0: PE1
1: PE4

Bit 5 **STPIPS**: STPI input source pin selection
0: PE2
1: PE3

Bit 4~3 **CTCK01PS~CTCK00PS**: CTCK0 input source pin selection
00: PE7
01: PF3
10: PA6
11: PE7

Bit 2~0 **RX22PS~RX20PS**: RX2/TX2 input source pin selection
000: PA5
001: PB3
010: PE5
011: PF7
100: PG0
101: PA5
110: PA5
111: PA5

• IFS3 Register

Bit	7	6	5	4	3	2	1	0
Name	IICSDA3PS	IICSDA2PS	IICSDA1PS	IICSDA0PS	IICSCL3PS	IICSCL2PS	IICSCL1PS	IICSCL0PS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~4 **IICSDA3PS~IICSDA0PS**: IICSDA input source pin selection
0000: PA2
0001: PA1
0010: PB0
0011: PB2
0100: PC6
0101: PE4
0110: PF0
0111: PF6
1000: PF7
1001: PG2
1010: PA2
1011: PA2
1101: PA2
1110: PA2
1111: PA2

Bit 3~0 **IICSCL3PS~IICSCL0PS**: IICSCL input source pin selection
0000: PA0
0001: PA7
0010: PB1
0011: PB3
0100: PC7
0101: PE5
0110: PF1
0111: PF7
1000: PF6
1001: PG3
1010: PA0
1011: PA0
1101: PA0
1110: PA0
1111: PA0

• **IFS4 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	INT11PS	INT10PS	SCSB2PS	SCSB1PS	SCSB0PS
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

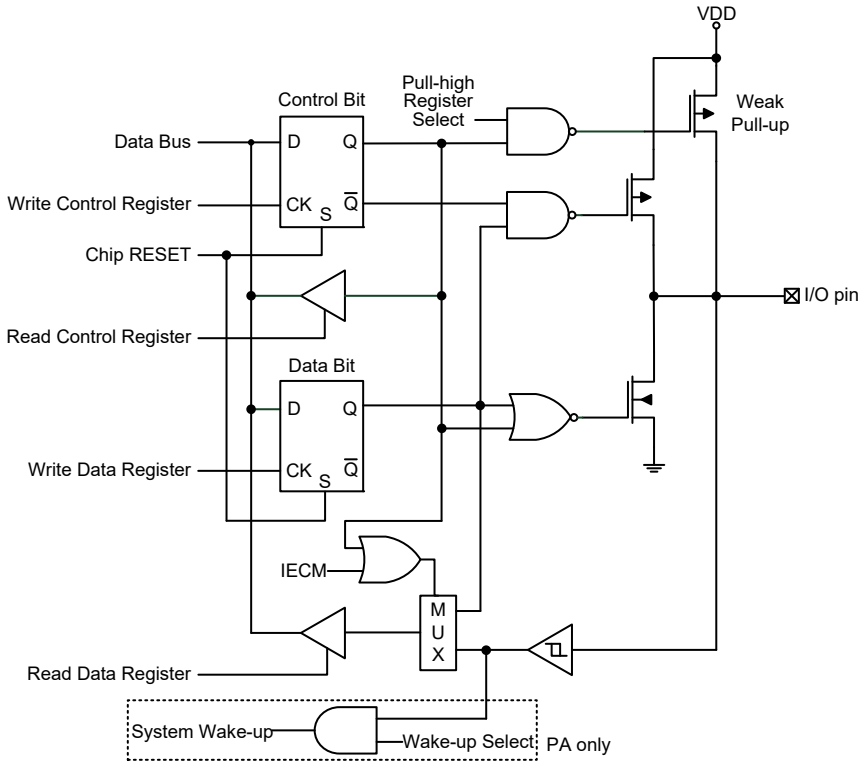
Bit 7~5 Unimplemented, read as "0"

Bit 4~3 **INT11PS~INT10PS**: INT1 input source pin selection
00: PB7
01: PE7
10: PB3
11: PB3

Bit 2~0 **SCSB2PS~SCSB0PS**: $\overline{SCS}$ input source pin selection
000: PA2
001: PA3
010: PE7
011: PF3
100: PF7
101: PG4
110: PB3
111: PE3

I/O Pin Structures

The accompanying diagram illustrates the internal structure of the I/O logic function. As the exact logical construction of the I/O pin will differ from this drawing, it is supplied as a guide only to assist with the functional understanding of the I/O logic function. 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 read data from I/O pins, which is specially designed for the IEC 60730 self-diagnostic test on the I/O function and A/D paths. After the self-diagnostic test on the I/O function and A/D paths are completed, users must disable the READ PORT function immediately. In cases other than the I/O function and A/D path self-diagnostic test, it is strongly recommended to disable the READ PORT function to avoid affecting other peripheral functions and causing unexpected consequences.

There is a register, IECC, which is used to control the READ PORT 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 reading path is from the I/O pins. The value on the corresponding pins will be passed to the accumulator ACC when the read port instruction "mov a, Px" is executed, where the "x" stands for the corresponding I/O port name. 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 or I/O pins. 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

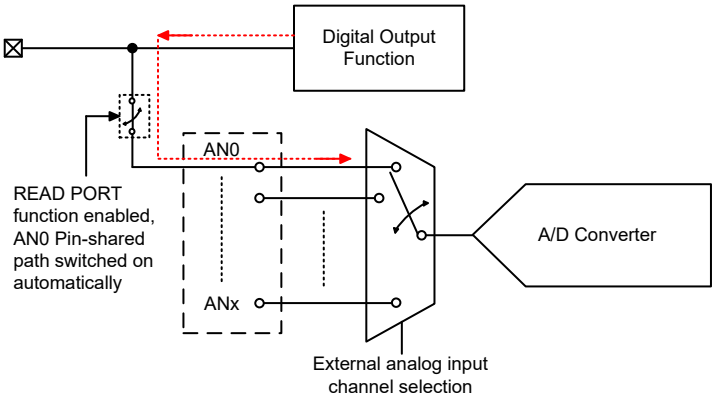
Bit 7~0 **IECS7~IECS0**: READ PORT function enable control bit 7 ~ bit 0
11001010: IECM=1 – READ PORT function is enabled
Others: IECM=0 – READ PORT function is disabled

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	
A/D Function	0			

Note: The value on 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, 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. As shown in the following 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 function. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the Port A pins. Single or multiple pins on Port A can be setup to have this function.

Timer Modules – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions the device includes several Timer Modules, generally abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

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

Introduction

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

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

TM Function Summary

TM Operation

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

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTnCK2~xTnCK0 bits in the xTMn control registers, where “x” stands for C, S or P type TM and “n” stands for the specific TM serial number. For STM there is no serial number “n” in the relevant pin, register and control bit names since there are only an STM in the device. The clock source can be a ratio of the system clock, f_{SYS}, or the internal high clock, f_H, the f_{SUB} clock source or the external xTCKn pin. The xTCKn pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

TM Interrupts

The Compact, 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 pins.

TM External Pins

Each of the TMs, irrespective of what type, has one TM input pin, with the label xTCKn. The xTMn input pin, xTCKn, is essentially a clock source for the xTMn and is selected using the xTCK2~xTCK0 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. For the Standard and Periodic Type TMs, the xTCKn pin is also used as the external trigger input pin in single pulse output mode.

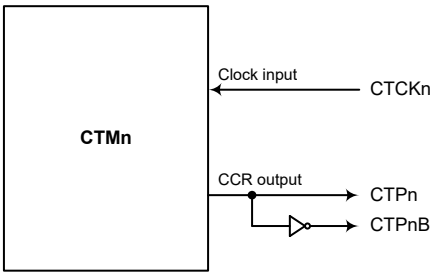
The Standard and Periodic Type TMs have another input pin, xTPnI, which is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the xTnIO1~xTnIO0 bits in the xTMnC1 register. There is another capture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

The TMs each have two output pins, with the label xTPn and xTPnB. When the TM is in the Compare Match Output Mode, 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 pins are also the pins where the TM generates the PWM output waveform.

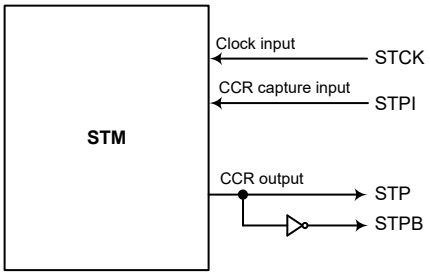
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using relevant pin-shared function selection register. The details of the pin-shared function selection are described in the pin-shared function section.

CTM		STM		PTM	
Input	Output	Input	Output	Input	Output
CTCK0 CTCK1 CTCK2	CTP0, CTP0B CTP1, CTP1B CTP2, CTP2B	STCK, STPI	STP, STPB	PTCK0, PTP0I PTCK1, PTP1I PTCK2, PTP2I	PTP0, PTP0B PTP1, PTP1B PTP2, PTP2B

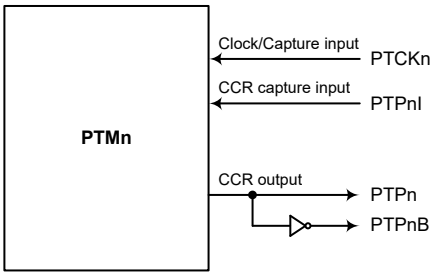
TM External Pins



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



STM Function Pin Block Diagram

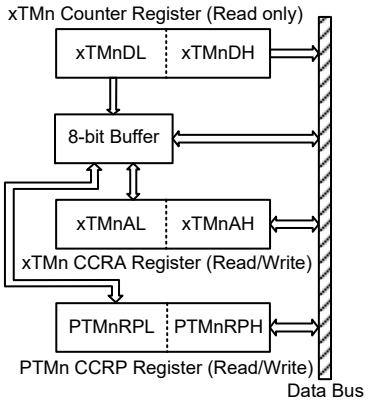


PTM Function Pin Block Diagram (n=0~2)

Programming Considerations

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

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



The following steps show the read and write procedures:

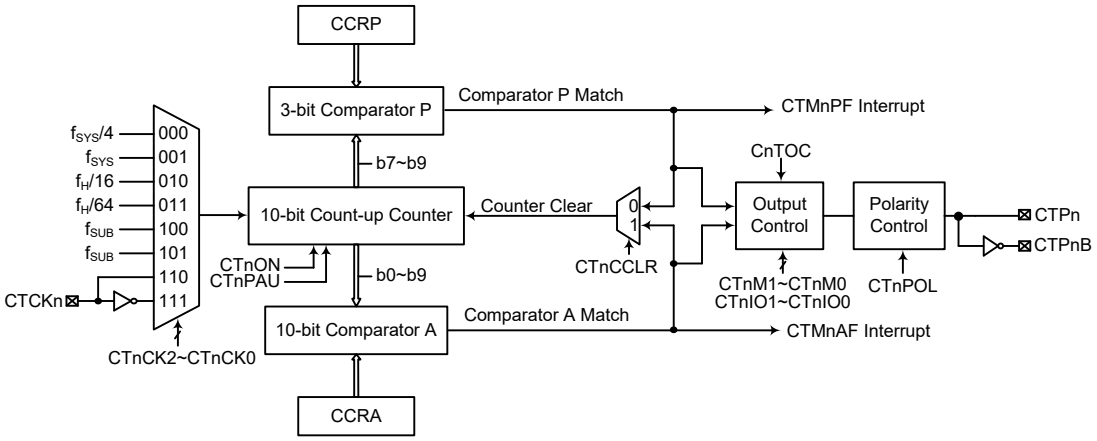
- Writing Data to CCRA or CCRP
  - Step 1. Write data to Low Byte xTMnAL or PTMnRPL
    - Note that here data is only written to the 8-bit buffer.

- Step 2. Write data to High Byte xTMnAH 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 or CCRP
  - Step 1. Read data from the High Byte xTMnDH, xTMnAH 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 or PTMnRPL
    - This step reads data from the 8-bit buffer.

Compact Type TM – CTM

The Compact TM type contains three operating modes, which are Compare Match Output, Timer/ Event Counter and PWM Output modes. The Compact TMs can also be controlled with an external input pin and can drive two external output pins.



Note: The CTMn external pins are pin-shared with other functions, therefore before using the CTMn function the relevant pin-shared function registers must be set properly to enable the CTMn pin function. The CTCKn pin, if used, must also be set as an imput by setting the corresponding bits in the port control register.

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

Compact Type TM Operation

The size of Compact TM is 10-bit wide ande 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 3 bits in the counter while the CCRA is the 10 bits and therefore compares with all counter bits.

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

Compact Type TM Register Description

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

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

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

- **CTMnC0 Register**

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

Bit 7 **CTnPAU**: CTMn counter pause control
0: Run
1: Pause

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

Bit 6~4 **CTnCK2~CTnCK0**: CTMn 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: CTCKn rising edge clock
111: CTCKn falling edge clock

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

Bit 3 **CTnON**: CTMn counter on/off control
0: Off
1: On

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

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

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

Comparator P Match Period=
000: 1024 CTMn clocks
001: 128 CTMn clocks
010: 256 CTMn clocks
011: 384 CTMn clocks
100: 512 CTMn clocks
101: 640 CTMn clocks
110: 768 CTMn clocks
111: 896 CTMn clocks

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

• CTMnC1 Register

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

Bit 7~6 **CTnM1~CTnM0**: CTMn operating mode selection
00: Compare Match Output Mode
01: Undefined
10: PWM Output Mode
11: Timer/Counter Mode

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

Bit 5~4 **CTnIO1~CTnIO0**: CTMn output pin function selection

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

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

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a compare match occurs from the Comparator A. The CTMn output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the CTMn

output pin should be setup using the CTnOC bit in the CTnMC1 register. Note that the output level requested by the CTnIO1 and CTnIO0 bits must be different from the initial value setup using the CTnOC bit otherwise no change will occur on the CTMn output pin when a compare match occurs. After the CTMn output pin changes state it can be reset to its initial level by changing the level of the CTnON bit from low to high.

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

Bit 3 **CTnOC**: CTMn CTPn output control

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

PWM Output Mode
0: Active low
1: Active high

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

Bit 2 **CTnPOL**: CTMn CTPn output polarity control
0: Non-invert
1: Invert

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

Bit 1 **CTnDPX**: CTMn PWM period/duty Control
0: CCRP – period, CCRA – duty
1: CCRP – duty; CCRA – period

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

Bit 0 **CTnCCLR**: CTMn counter clear condition selection
0: CTMn Comparatror P match
1: CTMn Comparatror A match

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

- **CTMnDL Register**

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

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

• **CTMnDH Register**

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

Bit 7~2 Unimplemented, read as "0"

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

• **CTMnAL Register**

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

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

• **CTMnAH Register**

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

Bit 7~2 Unimplemented, read as "0"

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

Compact Type TM Operating Modes

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

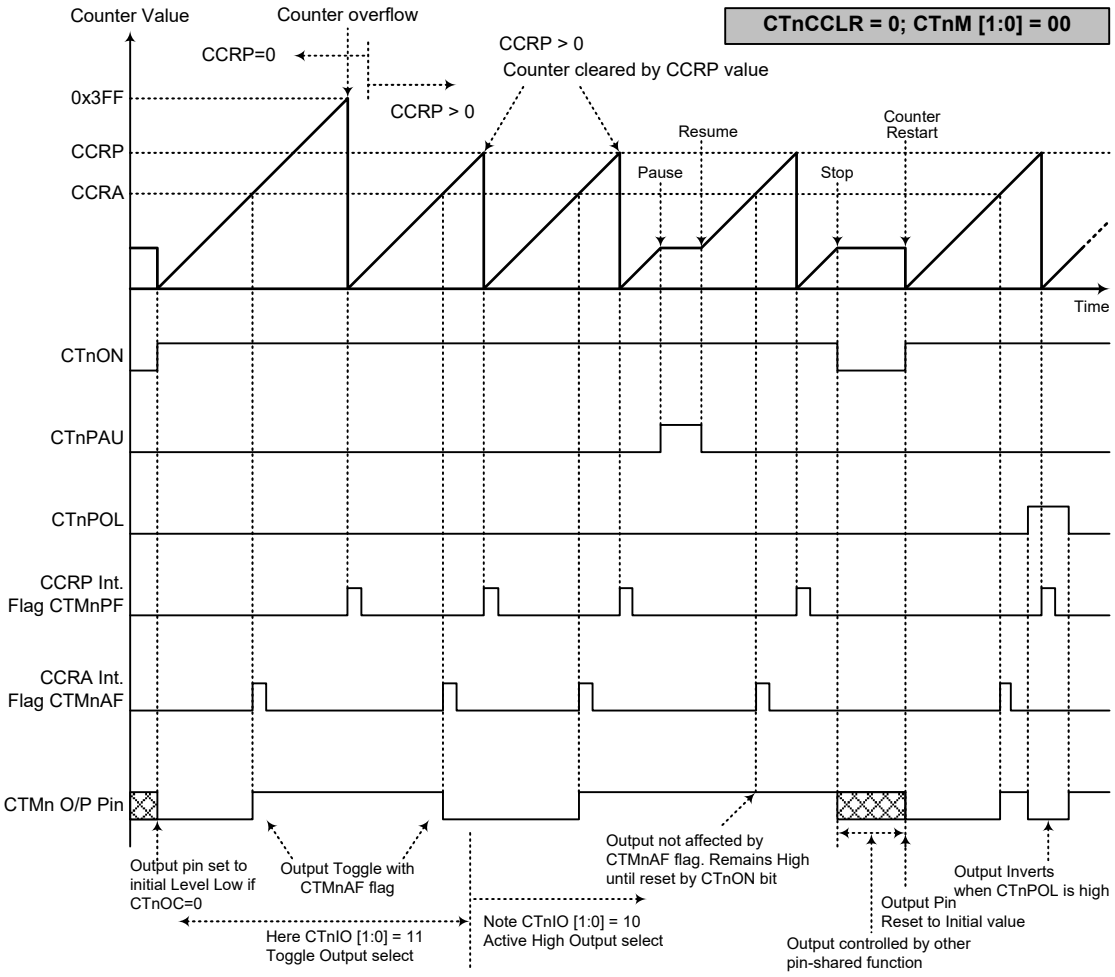
Compare Match Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTnMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMnAF and CTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

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

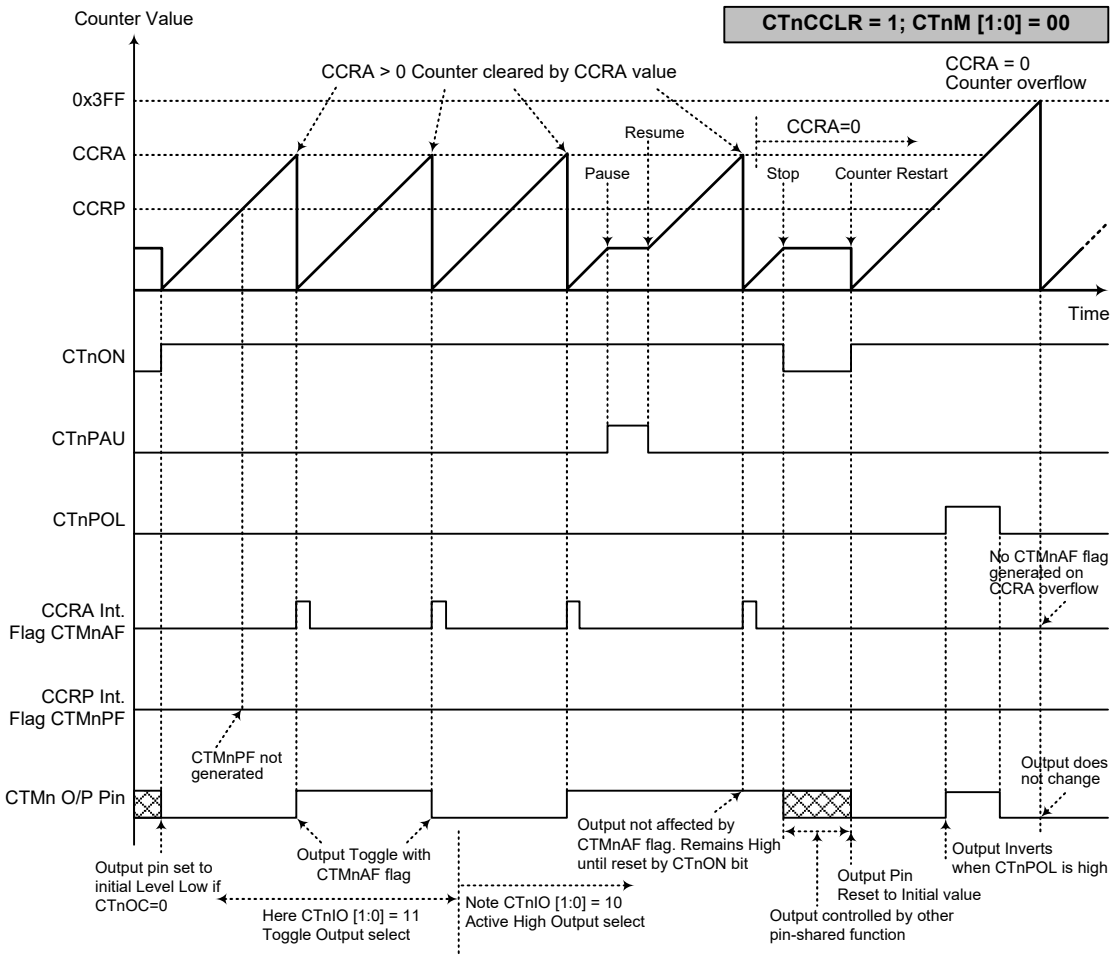
As the name of the mode suggests, after a comparison is made, the CTMn output pin will change state. The CTMn output pin condition however only changes state when a CTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMnPF interrupt

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



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

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



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

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

Timer/Counter Mode

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

PWM Output Mode

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

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

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

- **CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=0**

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

If f_{SYS}=8MHz, CTMn clock source is f_{SYS}/4, CCRP=4, CCRA=128,

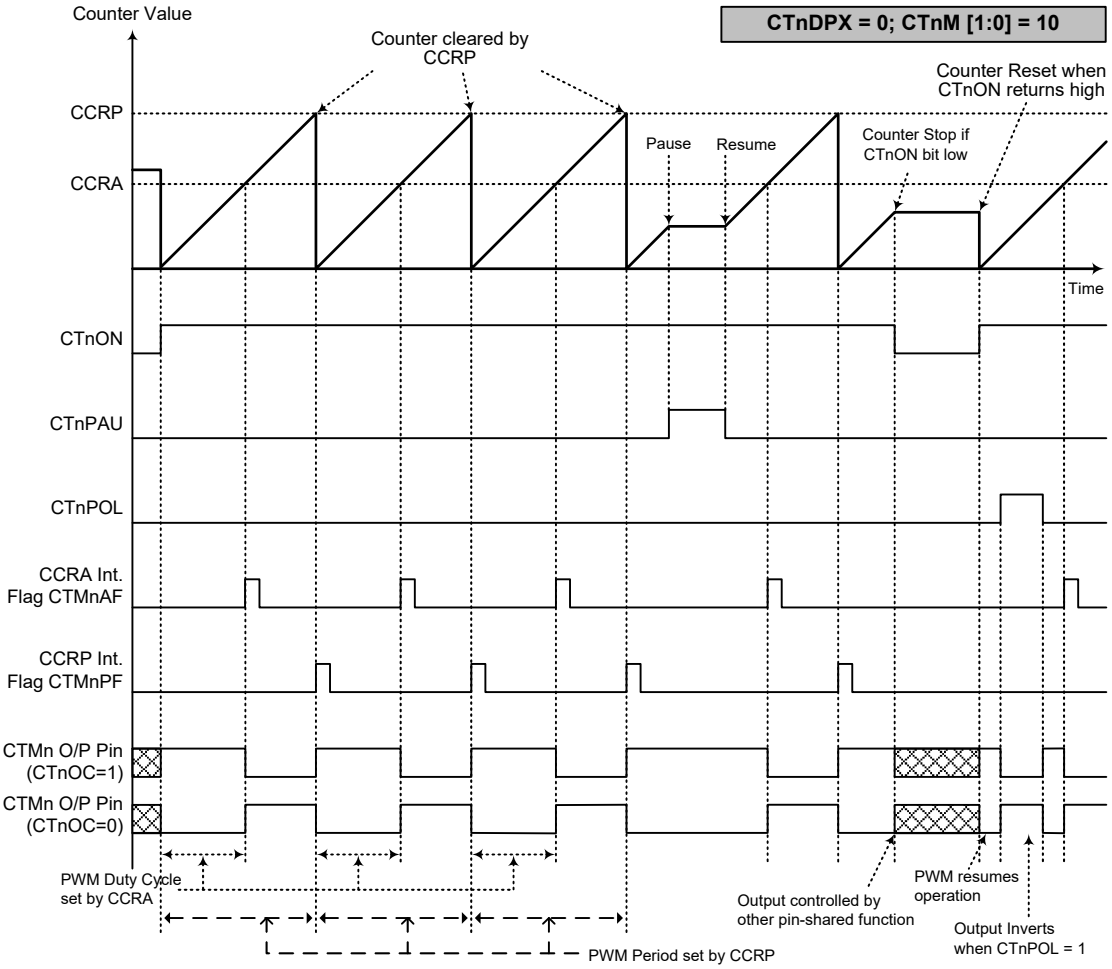
The CTMn PWM output frequency=(f_{SYS}/4)/(4×128)=f_{SYS}/2048=4kHz, duty=128/(4×128)=25%.

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%.

- **CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=1**

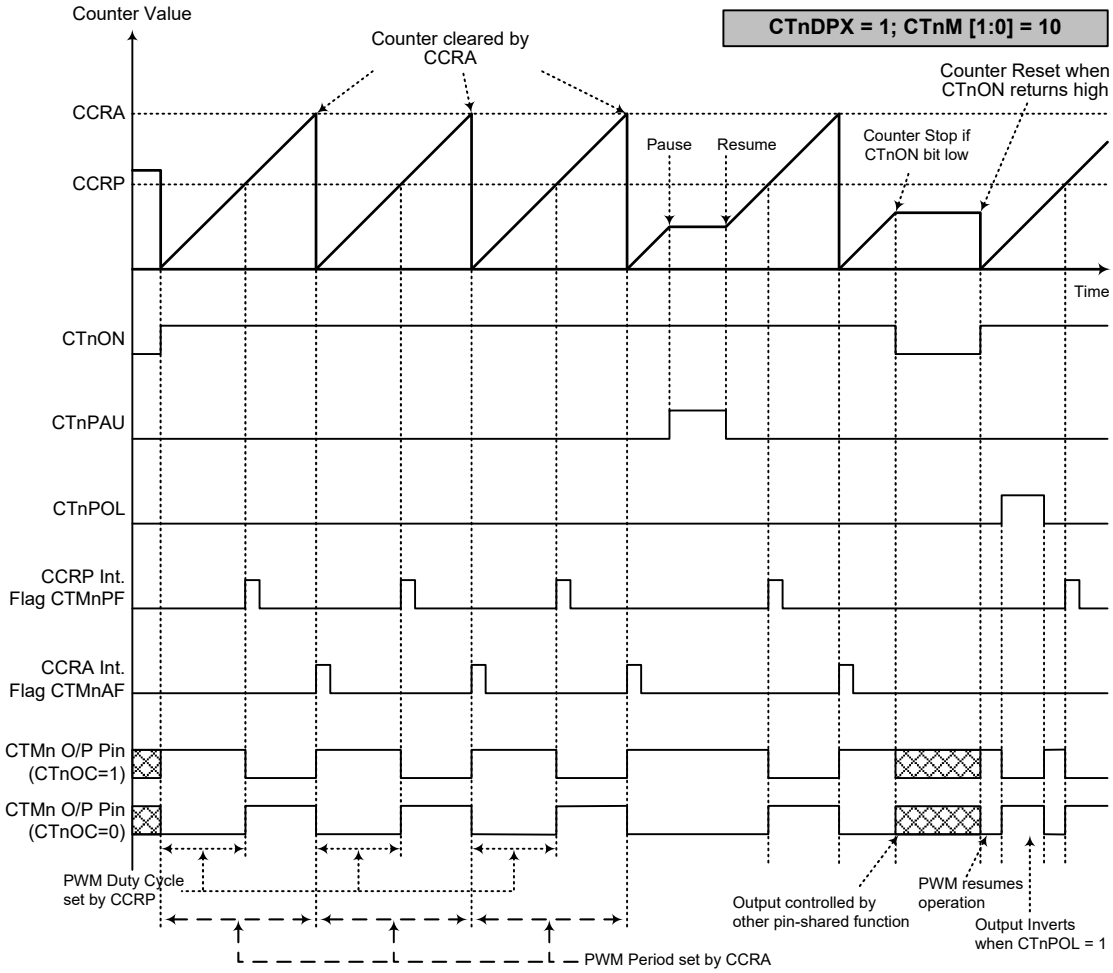
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

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



PWM Output Mode – CTnDPX=0 (n=0~2)

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

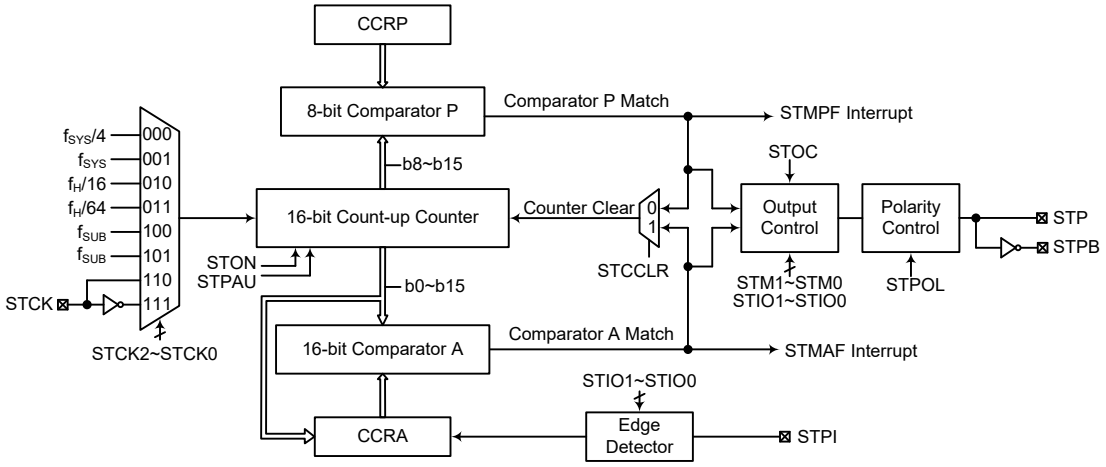


PWM Output Mode – CTnDPX=1 (n=0~2)

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

Standard Type TM – STM

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



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

16-bit Standard Type TM Block Diagram

Standard Type TM Operation

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

The only way of changing the value of the 16-bit counter using the application program, is to clear the counter by changing the STON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, an STM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

Standard Type TM Register Description

Overall operation of the Standard TM is controlled using a series of registers. A read only register pair exists to store the internal counter 16-bit value, while a read/write register pair exists to store the internal 16-bit CCRA value. The STMRP register is used to store the 8-bit CCRP bits. The remaining two registers are control registers which setup the different operating and control modes.

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

16-bit Standard TM Register List

- **STMC0 Register**

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

Bit 7 **STPAU**: STM counter pause control
0: Run
1: Pause
The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the STM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **STCK2~STCK0**: STM counter clock selection
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: STCK rising edge clock
111: STCK falling edge clock
These three bits are used to select the clock source for the STM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the "Operating Modes and System Clocks" section.

Bit 3 **STON**: STM counter on/off control
0: Off
1: On
This bit controls the overall on/off function of the STM. Setting the bit high enables the counter to run while clearing the bit disables the STM. Clearing this bit to zero will stop the counter from counting and turn off the STM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.
If the STM is in the Compare Match Output Mode, PWM Output Mode or Single Pulse Output Mode, then the STM output pin will be reset to its initial condition, as specified by the STOC bit, when the STON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

• **STMC1 Register**

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

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

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

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

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

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

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

Timer/Counter Mode
Unused

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

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

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

Bit 3 **STOC**: STM STP output control

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

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

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

Bit 2 **STPOL**: STM STP output polarity control
0: Non-invert
1: Invert

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

Bit 1 **STDPX**: STM PWM duty/period control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

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

Bit 0 **STCCLR**: STM counter clear condition selection
0: Comparator P match
1: Comparator A match

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

• STMDL Register

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

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

• STMDH Register

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

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

• **STMAL Register**

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

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

• **STMAH Register**

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

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

• **STMRP Register**

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

Bit 7~0 **D7~D0**: STM CCRP 8-bit Register, Compared with the STM Counter bit 15 ~ bit 8
Comparator P Match period =
0: 65536 STM clocks
1~255: (1~255)×256 STM clocks
These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight bits. The result of this comparison can be selected to clear the internal counter if the STCCLR bit is cleared to zero. Setting the STCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

Standard Type TM Operation Modes

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

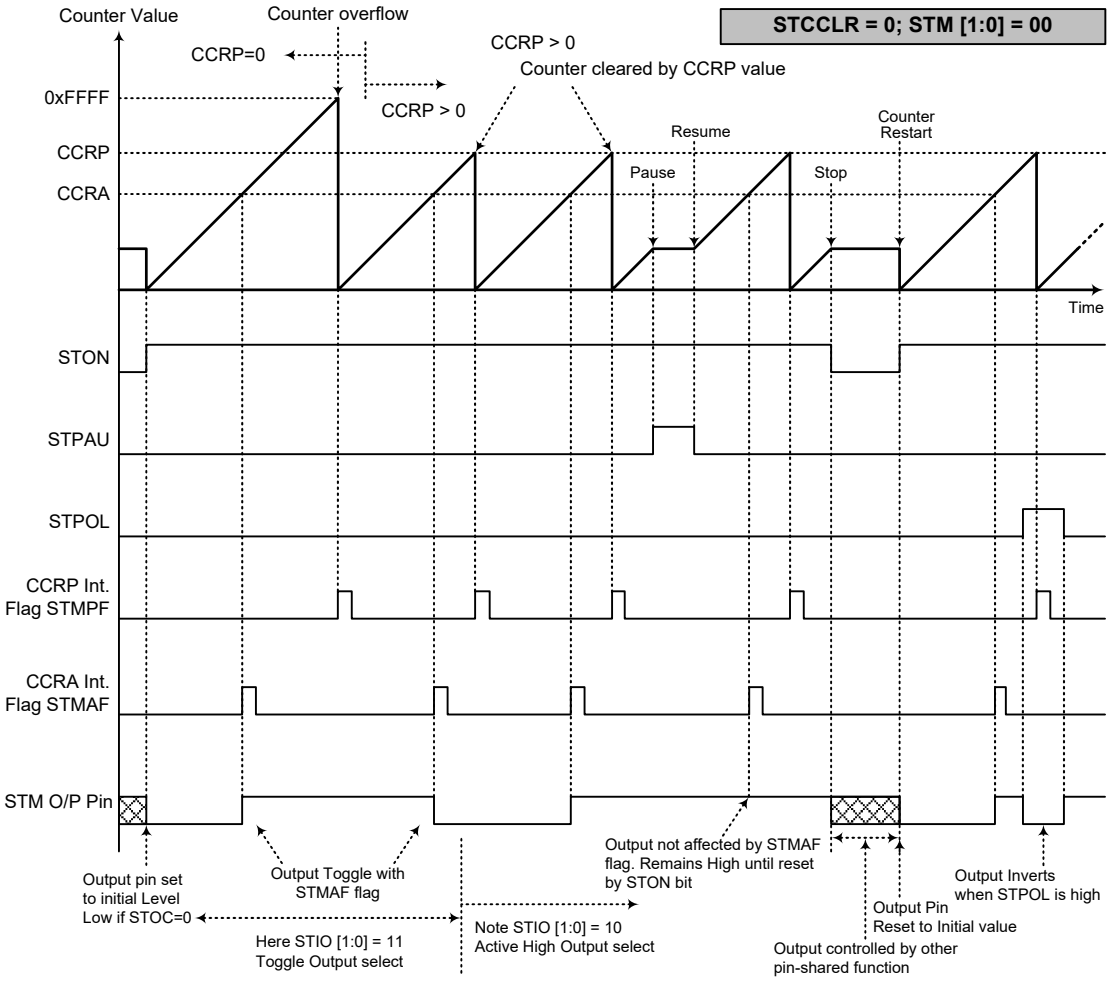
Compare Match Output Mode

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

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

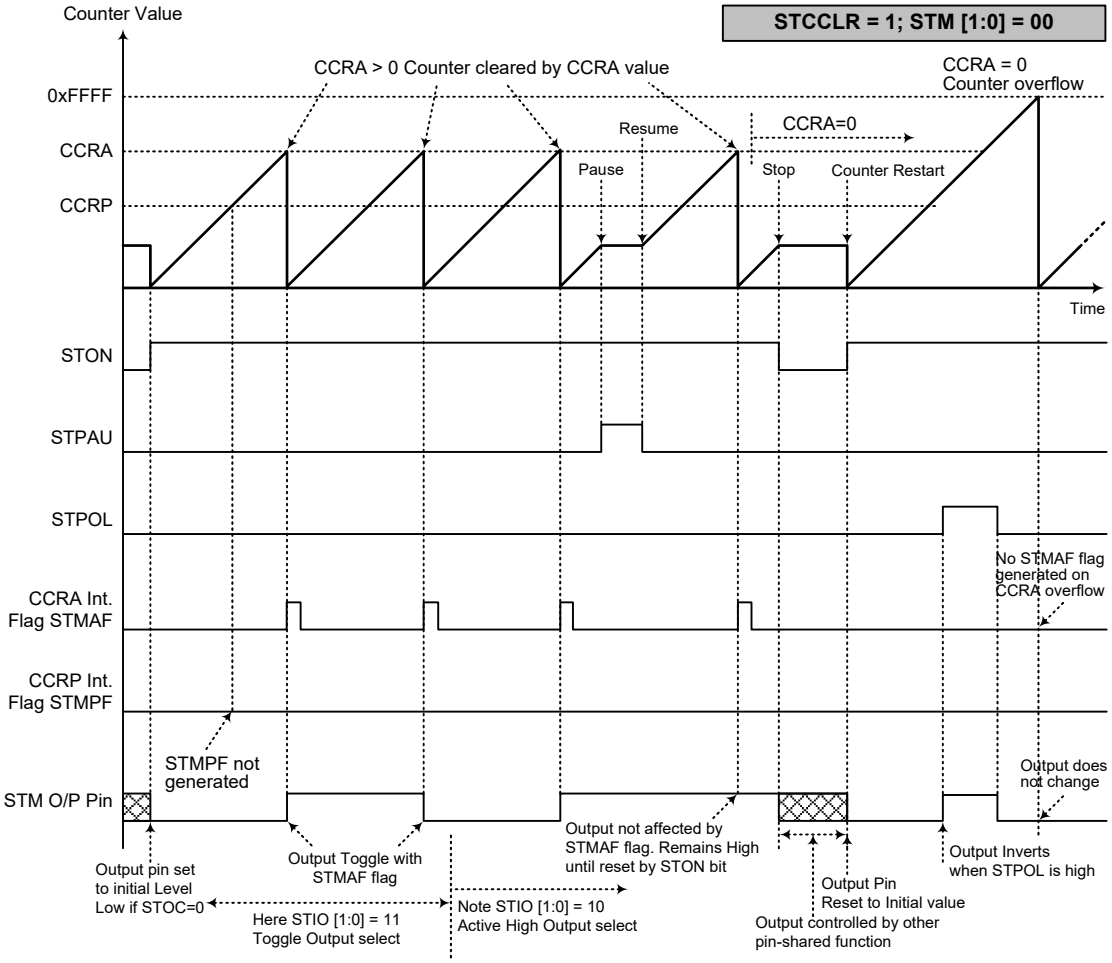
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 16-bit, FFFF Hex, value, however here the STMAF interrupt request flag will not be generated.

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



Compare Match Output Mode – STCCLR=0

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



Compare Match Output Mode – STCCLR=1

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

2. The STM output pin is controlled only by the STMAF flag
3. The output pin is reset to its initial state by an STON bit rising edge
4. An STMPF flag is not generated when STCCLR=1

Timer/Counter Mode

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

PWM Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively. The PWM function within the STM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the STM output pin, a square waveform AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output mode, the STCCLR bit has no effect on the PWM operation. Both of the CCRP and CCRA registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either period or duty cycle is determined using the STDPX bit in the STMC1 register. The PWM waveform period frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

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

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

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

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

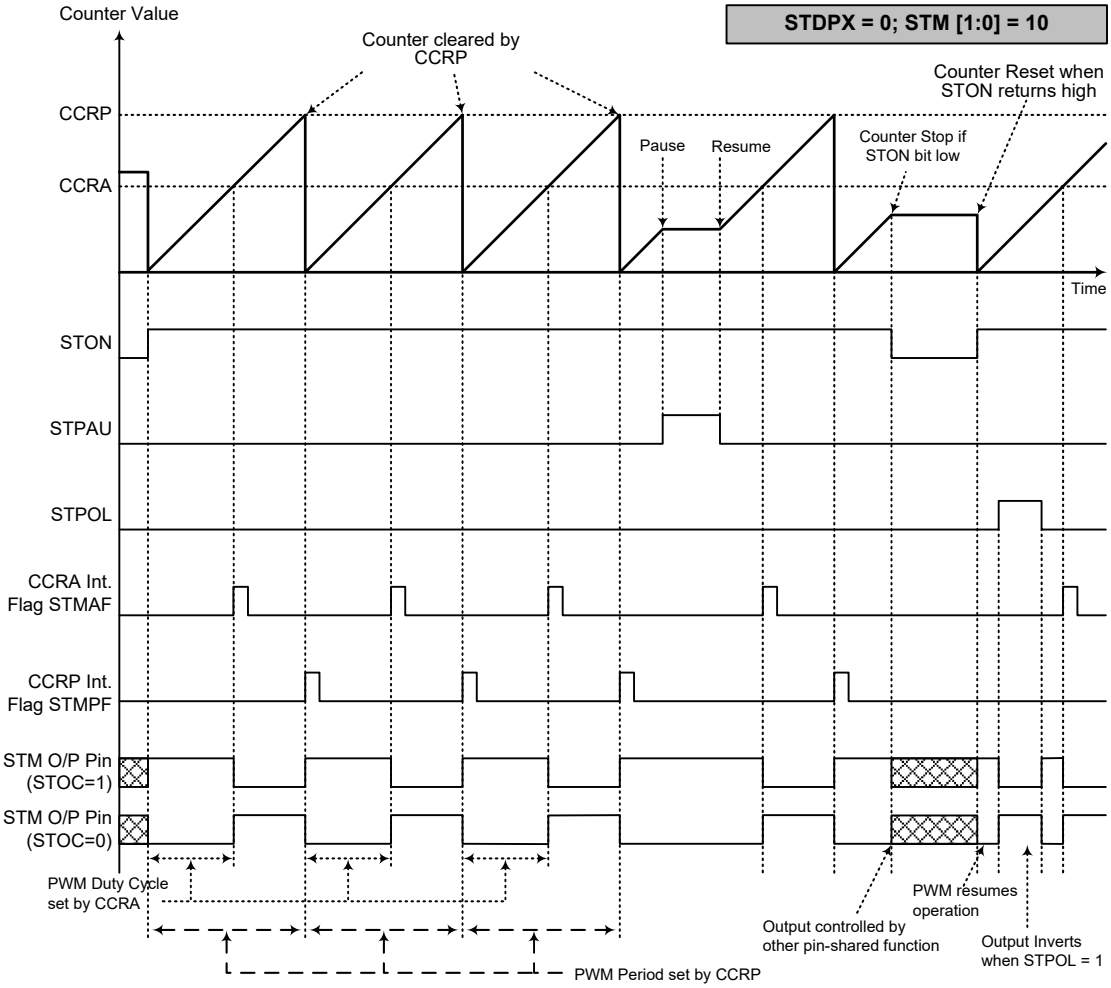
The STM PWM output frequency=$(f_{SYS}/4)/(2\times256)=f_{SYS}/2048$=4kHz, duty=128/(2×256)=25%.

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

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

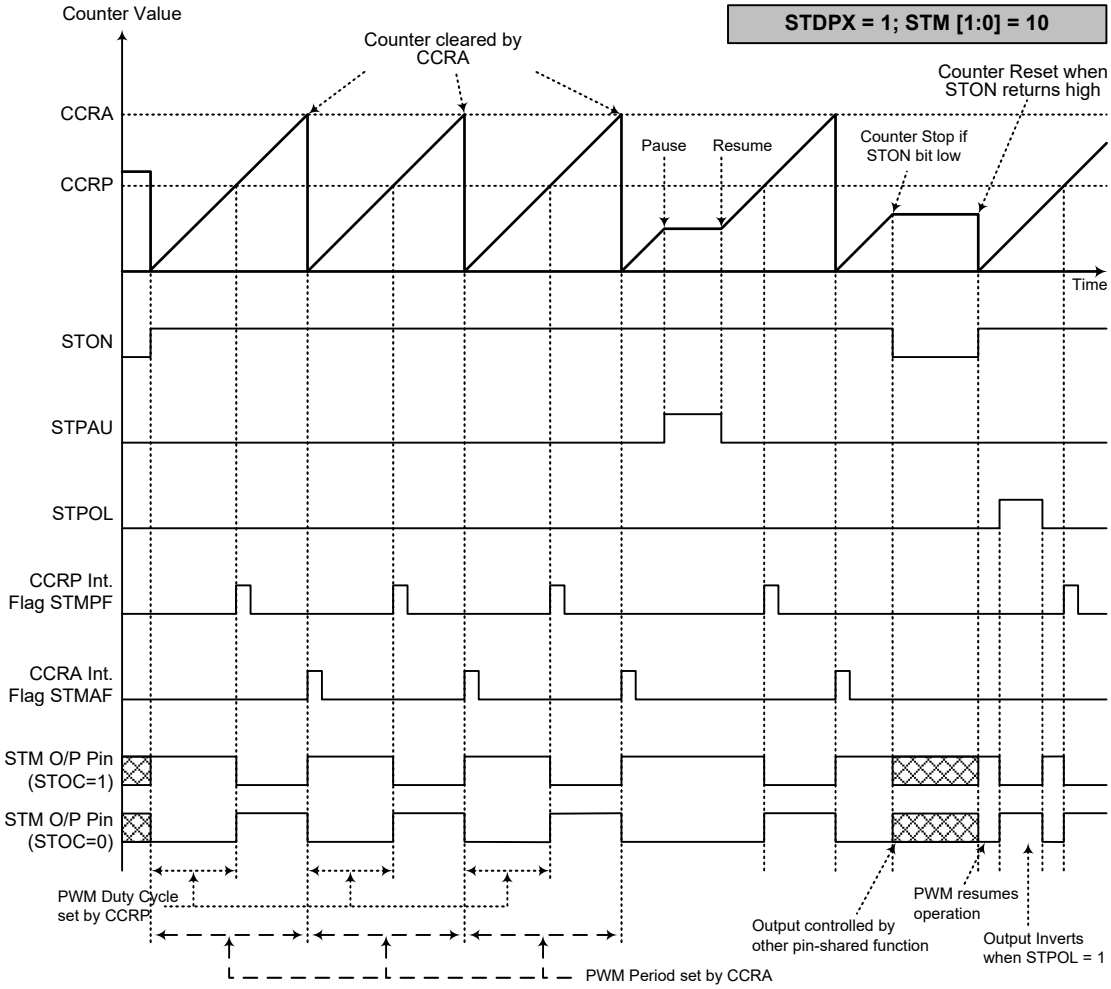
CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

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



PWM Output Mode – STDPX=0

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



PWM Output Mode – STDPX=1

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

2. A counter clear sets the PWM Period

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

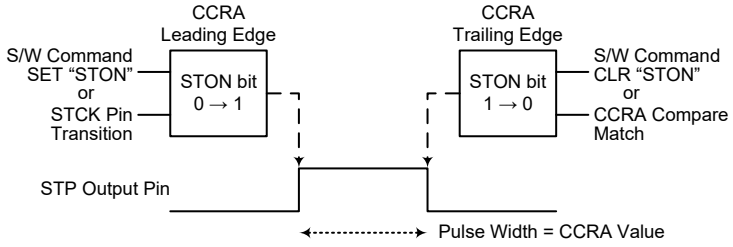
4. The STCCLR bit has no influence on PWM operation

Single Pulse Output Mode

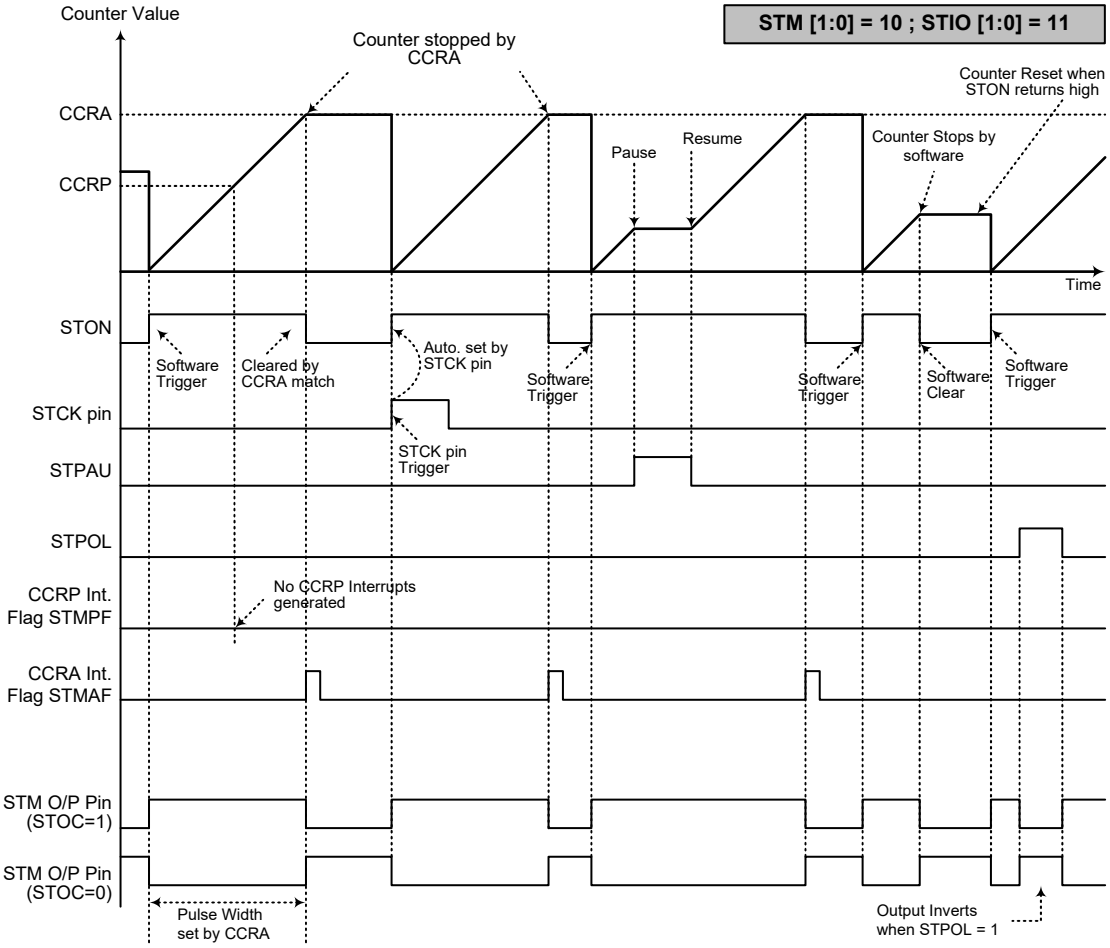
To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

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

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



Single Pulse Generation



Single Pulse Output Mode

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

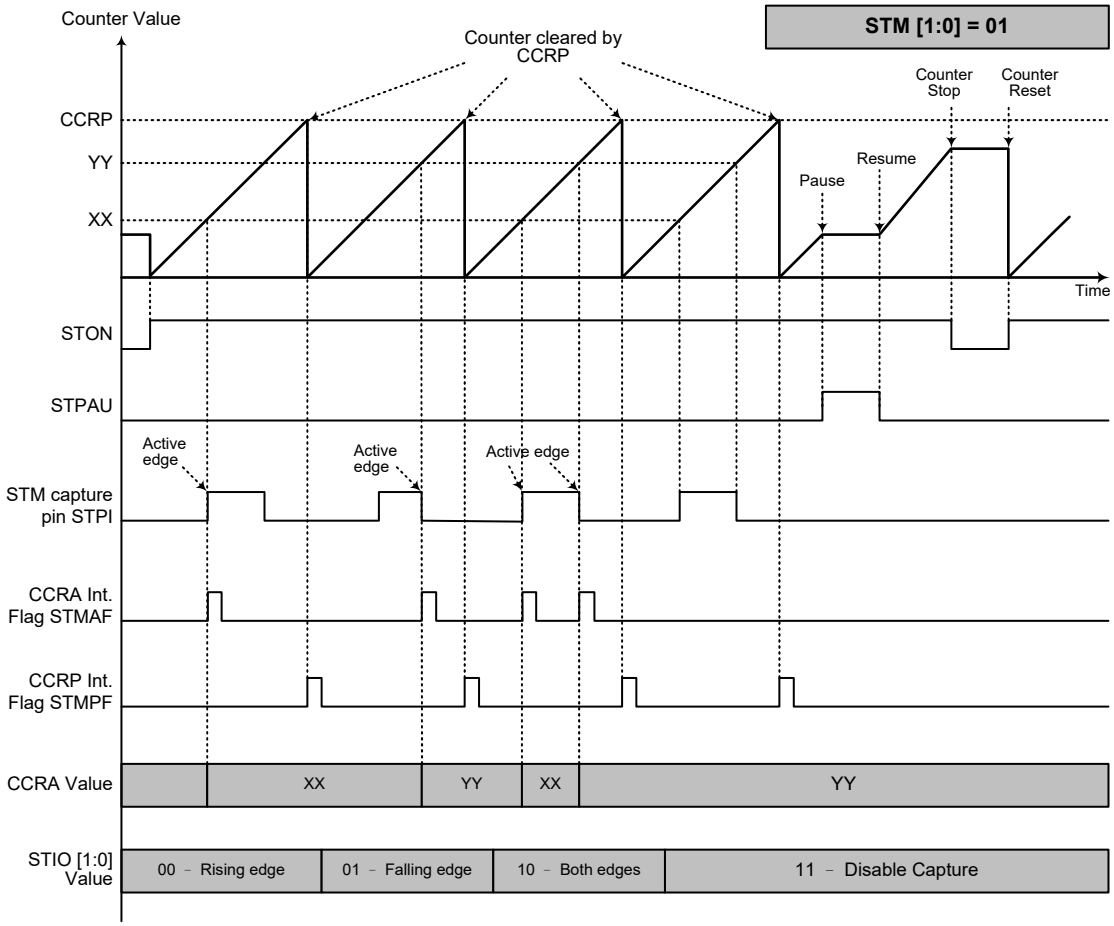
Capture Input Mode

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

When the required edge transition appears on the STPI pin the present value in the counter will be latched into the CCRA registers and an STM interrupt generated. Irrespective of what events occur on the STPI pin the counter will continue to free run until the STON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, an STM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The STIO1 and STIO0 bits can select the active trigger edge on the STPI pin to be a rising edge, falling edge or both edge types. If the STIO1 and STIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the STPI pin, however it must be noted that the counter will continue to run.

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

The STCCLR and STDPX bits are not used in this Mode.



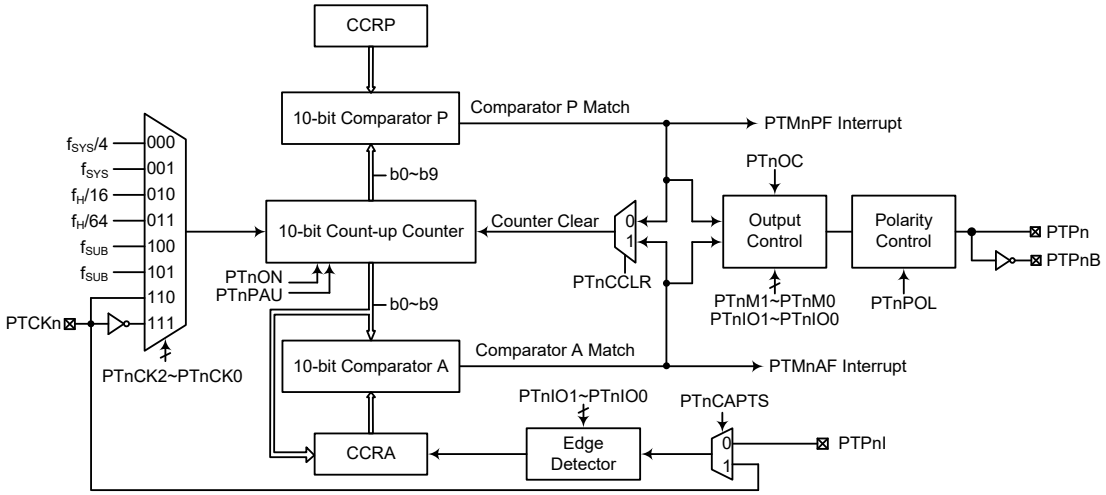
Capture Input Mode

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

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

Periodic Type TM – PTM

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



Note: The PTMn external pins are pin-shared with other functions, therefore before using the PTMn function the pin-shared function registers must 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.

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

Periodic Type TM Operation

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

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

Periodic Type TM Register Description

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

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

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

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

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

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

Bit 2~0 Unimplemented, read as "0"

• **PTMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	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
00: Input capture at rising edge of PTPnI or PTCKn
01: Input capture at falling edge of PTPnI or PTCKn
10: Input capture at rising/falling edge of PTPnI or PTCKn
11: Input capture disabled

Timer/Counter Mode
Unused

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

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

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

Bit 3 **PTnOC**: PTMn PTPn output control

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

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

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

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

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

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

Bit 0 **PTnCCLR**: PTMn counter clear condition selection
0: Comparator P match
1: Comparator A match

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

• PTMnDL Register

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

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

• PTMnDH Register

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

Bit 7~2 Unimplemented, read as “0”

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

• **PTMnAL Register**

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

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

• **PTMnAH Register**

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

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

• **PTMnRPL Register**

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

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

• **PTMnRPH Register**

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

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

Periodic Type TM Operation Modes

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

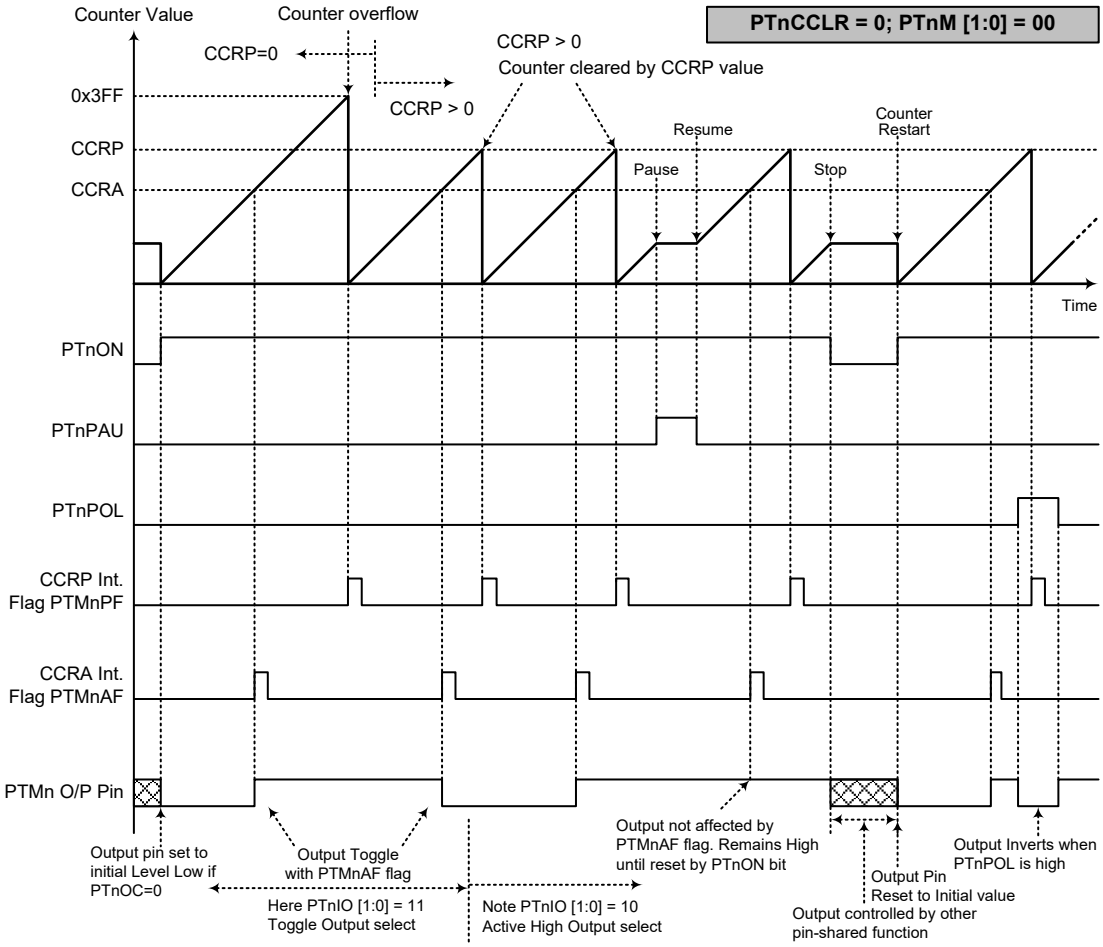
Compare Match Output Mode

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

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

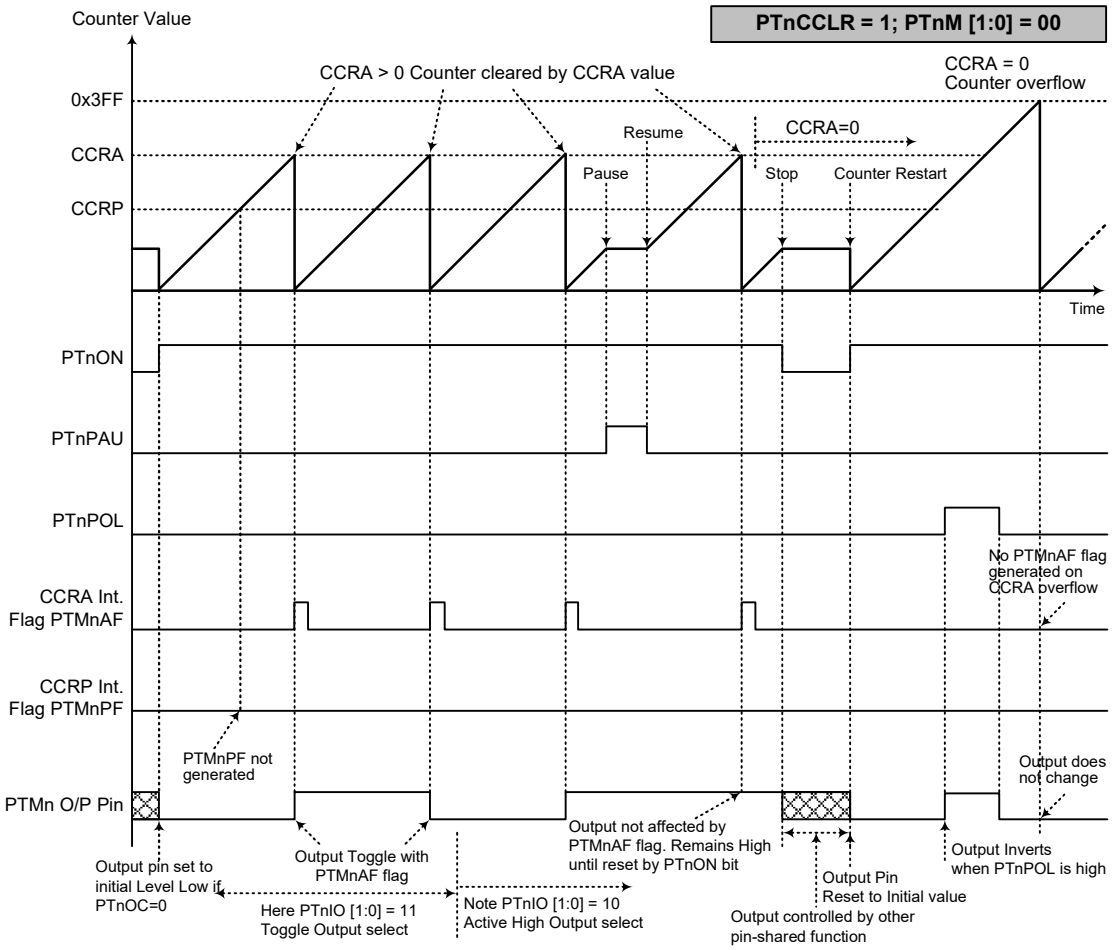
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the PTMnAF interrupt request flag will not be generated.

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



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 pin is controlled only by the PTMnAF flag
3. The output pin is reset to its initial state by a PTnON bit rising edge



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

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

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

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

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

Timer/Counter Mode

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

PWM Output Mode

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

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

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

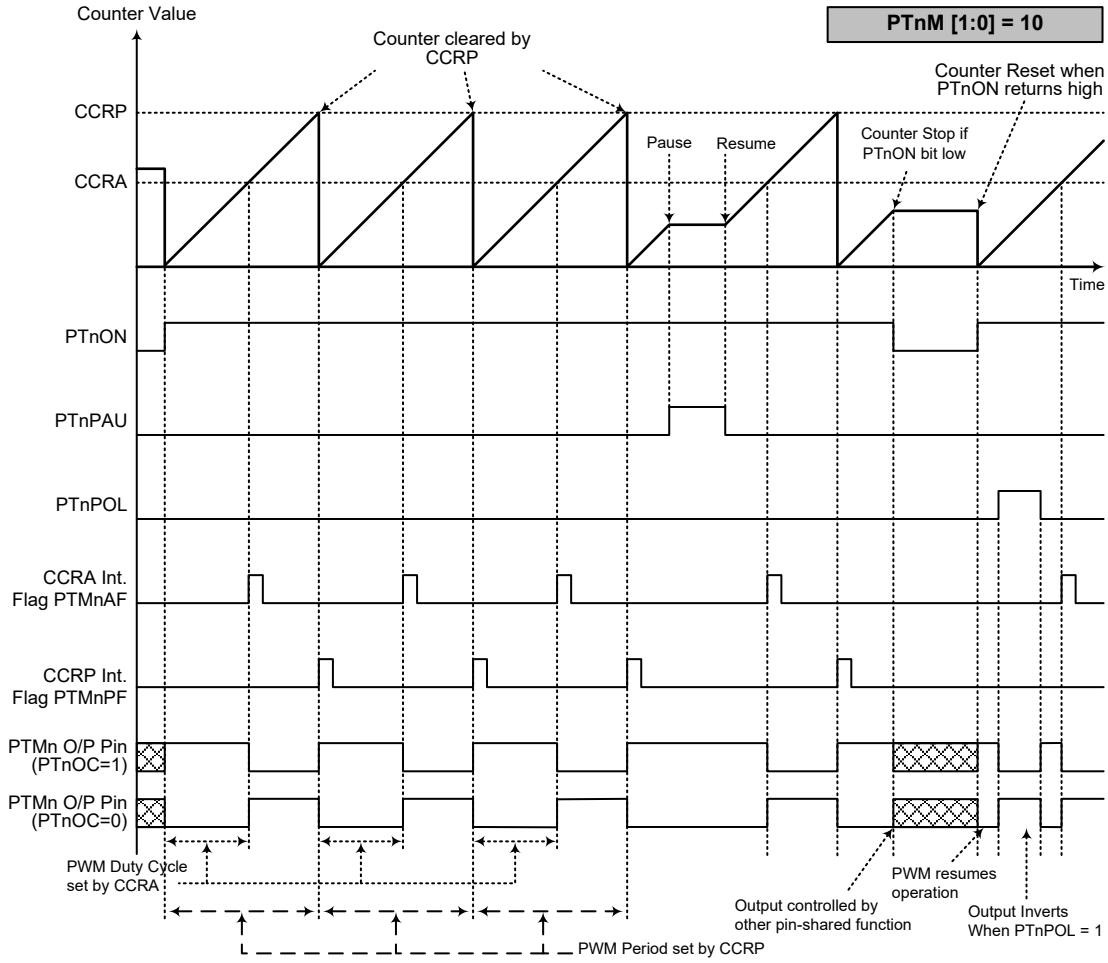
- **10-bit PTMn, PWM Output Mode, Edge-aligned Mode**

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

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

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

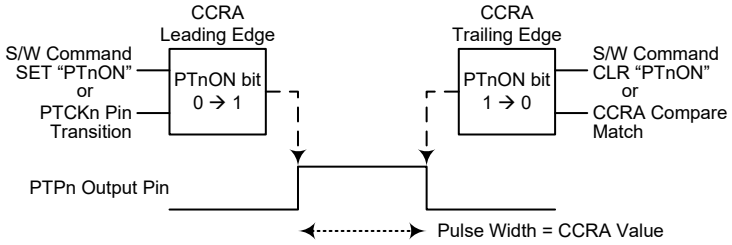
4. The PTnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

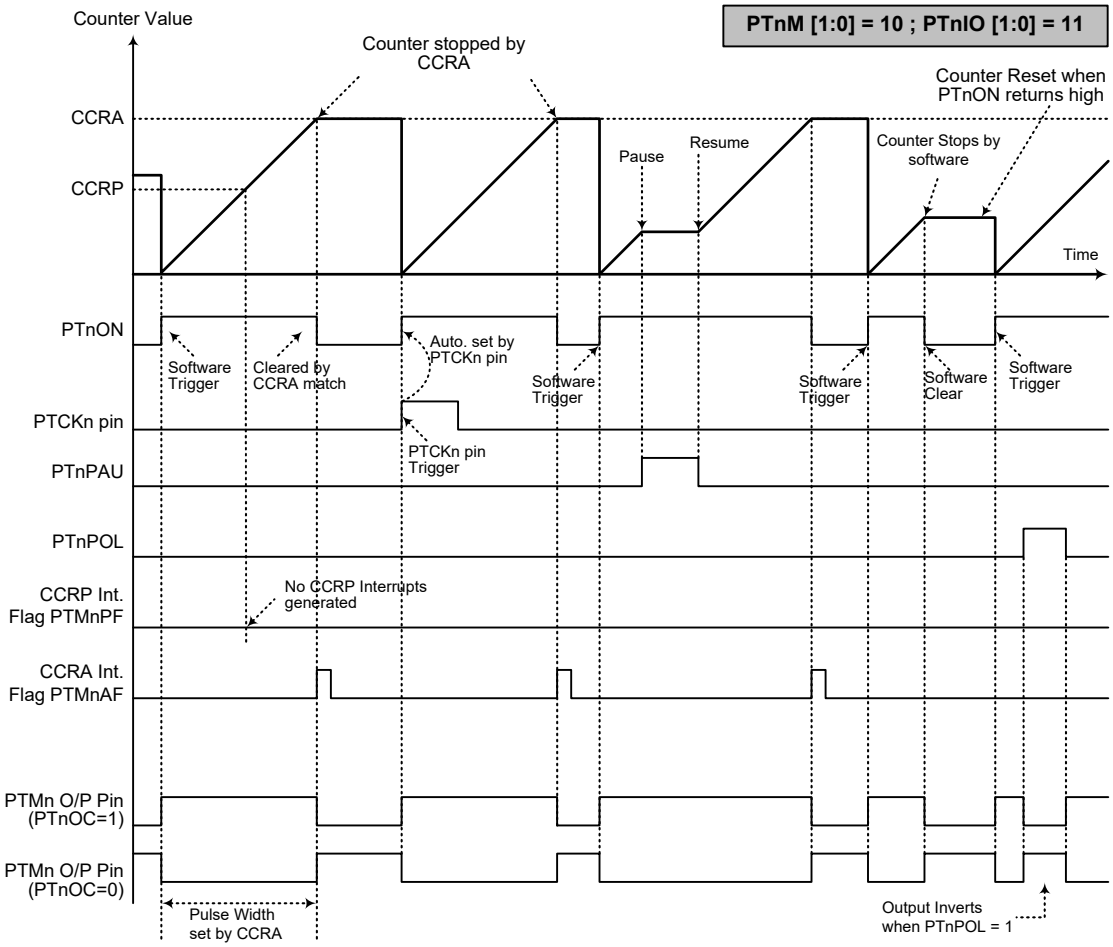
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

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

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



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 triggered by the PTCKn pin or by setting the PTnON bit high

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

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

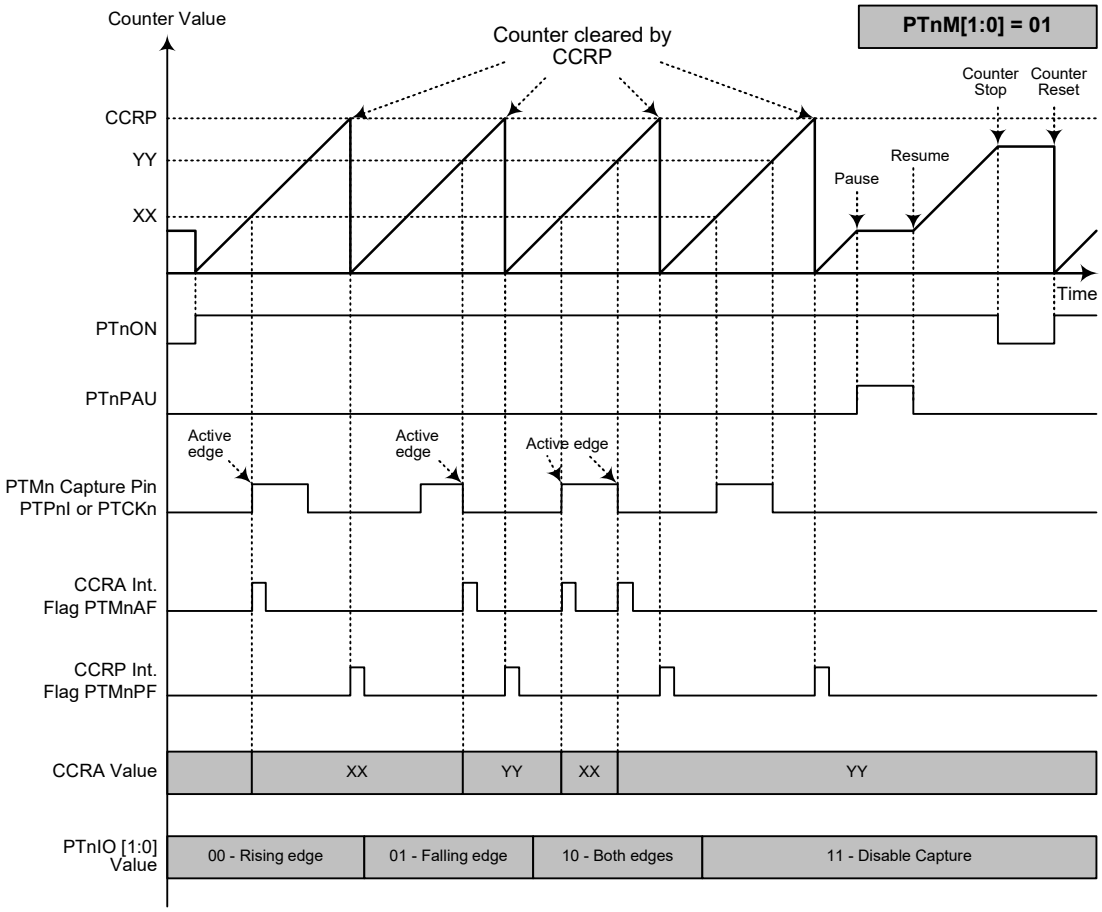
Capture Input Mode

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

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

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 PTMAnF 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.

The PTnCCLR, PTnOC and PTnPOL bits are not used in this Mode.

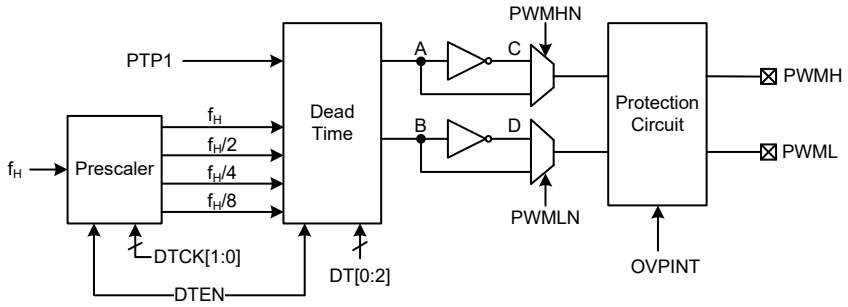


Capture Input Mode (n=0~2)

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

Complementary PWM Output with Dead Time

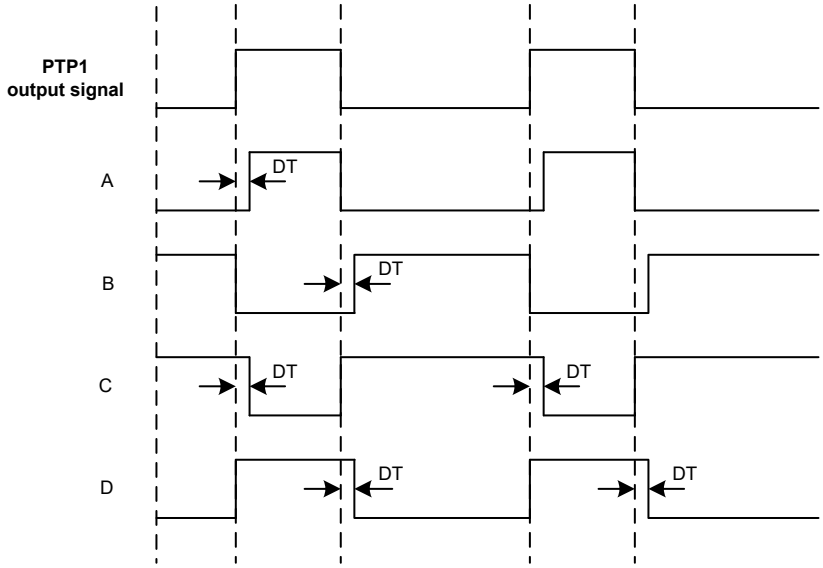
The device provides a complementary output pair of signals which can be used as a PWM driver signal. The PWM signal is sourced from the PTM1 PTP1 output which is an active high signal. A dead time will be inserted into the PTM1 PTP1 output signal to prevent excessive DC currents. In addition to register configuration, the complementary PWM output can also be stopped by an OVP condition occurrence, when such condition occurs and the corresponding control bit in the OCVPC register is enabled, the PWM output will stop and the PWM output pair status will be forced to a certain level determined by the PWMHOPS and PWMLOPS bits.



Complementary PWM Output with Dead Time Block Diagram

Dead Time Insertion

The complementary PWM output circuit provides a dead time insertion function. By setting the DTEN bit in the CPR register, the dead time generator and prescaler will be enabled. The clock source of the prescaler originates from the internal clock f_H and the division ratio is determined by the DTCK[1:0] bits. When the related register bits are properly configured, a dead time, which is programmable using the DT[2:0] bits in the CPR register, will be inserted to prevent excessive DC currents. The dead time will be inserted whenever the rising edge of the dead time generator input signal, namely the PTM1 PTP1 output signal, occurs.



Complementary PWM Output with Dead Time Control

Complementary PWM Registers

The complementary PWM output function can be controlled using internal registers. The CPR register is used to control the dead time function enable/disable, PWMH/PWML inverse signal selection, dead time prescaler selection and dead time selection. The OCVPC register is used to control the protection circuit and determine the PWM output pair status when the complementary PWM output circuit is stopped.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CPR	DTEN	PWMHN	PWMLN	DTCK1	DTCK0	DT2	DT1	DT0
OCPVC	—	—	—	—	PWMHOPS	PWMLOPS	D1	PWMOVEN

Complementary PWM Register List

• CPR Register

Bit	7	6	5	4	3	2	1	0
Name	DTEN	PWMHN	PWMLN	DTCK1	DTCK0	DT2	DT1	DT0
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 **DTEN**: Dead Time on/off control
0: Dead time & prescaler off
1: Dead time & Prescaler on
When this bit is cleared to zero, the PWMH and PWML status are determined by the PWMHOPS and PWMLOPS bits respectively.

Bit 6 **PWMHN**: PWMH inverse signal selection
0: PWMH=A
1: PWMH=C

Bit 5 **PWMLN**: PWML inverse signal selection
0: PWML=B
1: PWML=D

Bit 4~3 **DTCK1~DTCK0**: Dead time prescaler selection
00: $f_D=f_H$
01: $f_D=f_H/2$
10: $f_D=f_H/4$
11: $f_D=f_H/8$

Bit 2~0 **DT2~DT0**: Dead time selection
000: $[(1/f_D)-(1/f_H)]\sim(1/f_D)$
001: $[(2/f_D)-(1/f_H)]\sim(2/f_D)$
010: $[(3/f_D)-(1/f_H)]\sim(3/f_D)$
011: $[(4/f_D)-(1/f_H)]\sim(4/f_D)$
100: $[(5/f_D)-(1/f_H)]\sim(5/f_D)$
101: $[(6/f_D)-(1/f_H)]\sim(6/f_D)$
110: $[(7/f_D)-(1/f_H)]\sim(7/f_D)$
111: $[(8/f_D)-(1/f_H)]\sim(8/f_D)$
Note: $t_D=1/f_D$.

• **OCVPC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PWMHOPS	PWMLOPS	D1	PWMOVEN
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **PWMHOPS**: PWMH output status when complementary PWM output is stopped
0: Output 0
1: Output 1
When the complementary PWM output circuit is stopped by clearing the DTEN bit to zero, or by an OVP condition occurrence, the PWMH output status will be forced to output 1 if the PWMHOPS bit is set to "1", otherwise the output status will be forced to output 0 if this bit is cleared to "0". Note that configuring this bit has no effect when the complementary PWM output circuit is in normal operation.

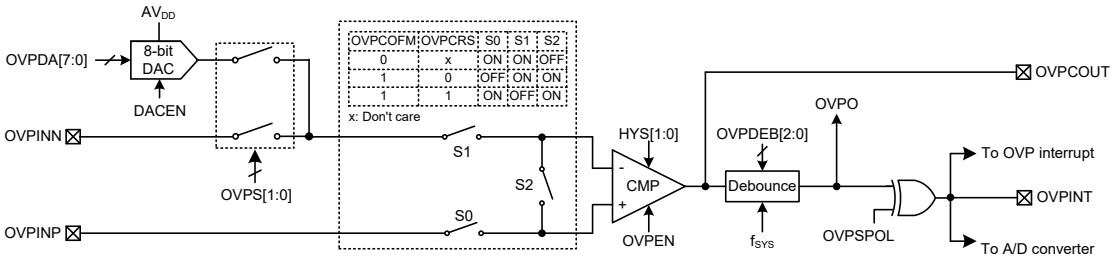
Bit 2 **PWMLOPS**: PWML output status when complementary PWM output is stopped
0: Output 0
1: Output 1
When the complementary PWM output circuit is stopped by clearing the DTEN bit to zero, or by an OVP condition occurrence, the PWML output status will be forced to output 1 if the PWMLOPS bit is set, otherwise the output status will be forced to output 0 if this bit is cleared to zero. Note that configuring this bit has no effect when the complementary PWM output circuit is in normal operation.

Bit 1 **D1**: Reserved bit, keep unchanged after power-on

Bit 0 **PWMOVEN**: PWM over voltage protection enable control
0: Disable
1: Enable
This bit is used to determine if an OVP condition occurrence will affect the PWM output circuit. If an OVP condition occurs and this bit is set high, the DTEN bit will be automatically cleared to zero by hardware to stop the complementary PWM output circuit. In this case, the PWMH and PWML status will be forced to a fixed high or low level determined by the PWMHOPS and PWMLOPS bits respectively.

Over Voltage Protection – OVP

The device includes an over voltage protection function which provides a protection mechanism for applications. To prevent the operating voltage from exceeding a specific level, the voltage on the OVPINP pin is compared with a reference voltage generated by an 8-bit D/A converter or an input voltage from the OVPINN pin. When an over voltage event occurs, the OVP will issue an output signal OVPO to indicate the source voltage is over specifications. An OVP interrupt will be generated if the corresponding interrupt control is enabled.



OVP Block Diagram

Over Voltage Protection Registers

Overall operation of the OVP function is controlled using several registers. One register is used to provide the reference voltages for the over voltage protection circuit. The remaining three registers are control registers which are used to control the OVP function, D/A converter, comparator debounce time, comparator hysteresis function together with the comparator input offset calibration.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OVPC0	OVPO	OVPSPOL	OVPEN	—	—	OVPDEB2	OVPDEB1	OVPDEB0
OVPC1	OVPCOUT	OVPCOFM	OVPCRS	OVPCOF4	OVPCOF3	OVPCOF2	OVPCOF1	OVPCOF0
OVPC2	DACEN	—	—	—	HYS1	HYS0	OVPS1	OVPS0
OVPDA	D7	D6	D5	D4	D3	D2	D1	D0

OVP Register List

- **OVPC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	OVPO	OVPSPOL	OVPEN	—	—	OVPDEB2	OVPDEB1	OVPDEB0
R/W	R	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7 **OVPO**: OVP comparator output bit after debounce
0: Positive input voltage < negative input voltage
1: Positive input voltage > negative input voltage

Bit 6 **OVPSPOL**: OVP debounded output signal polarity control
0: Non-invert
1: Invert
This bit determines the OVP interrupt occurrence condition when the OVPO bit changes state from low to high or high to low. Note that this bit is always effective even when OVPEN bit is 0. When OVPEN=0 and OVPSPOL=0, the OVPINT output is in a pull-low state, when OVPEN=0 and OVPSPOL=1, the OVPINT output is in a pull-high state.

Bit 5 **OVPEN**: OVP function enable control
0: Disable
1: Enable

If the OVPEN bit is cleared to 0, the over voltage protection function is disabled and no power will be consumed. This results in the comparator being switched off.

Bit 4~3 Unimplemented, read as "0"

Bit 2~0 **OVPDEB2~OVPDEB0**: OVP comparator output debounce time selection
000: Bypass, without debounce
001: (1~2)×t_{DEB}
010: (3~4)×t_{DEB}
011: (7~8)×t_{DEB}
100: (15~16)×t_{DEB}
101: (31~32)×t_{DEB}
110: (63~64)×t_{DEB}
111: (127~128) t_{DEB}
Note: t_{DEB}=1/f_{SYS}.

• OVPC1 Register

Bit	7	6	5	4	3	2	1	0
Name	OVPCOUT	OVPCOFM	OVPCRS	OVPCOF4	OVPCOF3	OVPCOF2	OVPCOF1	OVPCOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **OVPCOUT**: OVP comparator output bit before debounce
0: Positive input voltage < negative input voltage
1: Positive input voltage > negative input voltage
When the OVPEN bit is 0 the OVPCOUT output is in a pull-low state.

Bit 6 **OVPCOFM**: OVP comparator operating mode selection
0: Normal operating mode
1: Input offset voltage calibration mode
This bit is used to select the OVP comparator operating mode. To select the comparator input offset voltage calibration mode, the OVPCOFM bit must be set to 1. Refer to the "Comparator Input Offset Calibration" section for the detailed offset calibration procedures.

Bit 5 **OVPCRS**: OVP comparator input offset voltage calibration reference selection
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 4~0 **OVPCOF4~OVPCOF0**: OVP comparator input offset voltage calibration value
This 5-bit field is used to perform the comparator input offset voltage calibration operation and the value for the OVP comparator input offset voltage calibration can be restored into this bit field. More detailed information is described in the "Comparator Input Offset Calibration" section.

• OVPC2 Register

Bit	7	6	5	4	3	2	1	0
Name	DACEN	—	—	—	HYS1	HYS0	OVPS1	OVPS0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **DACEN**: OVP D/A converter enable control
0: Disable
1: Enable

Bit 6~4 Unimplemented, read as "0"

Bit 3~2 **HYS1~HYS0**: OVP comparator hysteresis voltage window control bits
Refer to "Over Voltage Protection Electrical Characteristics" table for detail.

Bit 1~0 **OVPS1~OVPS0**: OVP comparator negative input selection
00: No selection
01: OVPINN

10: D/A converter output voltage
11: No selection

• **OVPDA 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**: OVP D/A converter output voltage control bits
OVP D/A converter output $V_{OUT}=(AV_{DD}/256)\times D[7:0]$

Comparator Input Offset Calibration

As the OVP input pins are pin-shared with I/O or other pin functions, they should first be configured as the OVP inputs using the corresponding pin-share function control bits. Before offset calibration, the hysteresis voltage should be zero by setting the HYS1~HYS0 bits to 00. The procedures and settings of the comparator input offset calibration are shown as follows.

- Step 1

  Set OVPCOFM=1 and OVPCRS=1, the OVP will now operate in the comparator input offset voltage calibration mode (S0 and S2 on). To make sure the V_{OS} is as minimized 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 OVPCOF[4:0]=00000 then read the OVPCOUT bit.

- Step 3

  Increase the OVPCOF[4:0] value by 1 and then read the OVPCOUT bit.

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

  If the OVPCOUT bit state has changed, record the OVPCOF value as V_{OS1} and then go to Step 4.

- Step 4

  Set OVPCOF[4:0]=11111 and read the OVPCOUT bit.

- Step 5

  Decrease the OVPCOF[4:0] value by 1 and then read the OVPCOUT bit.

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

  If the OVPCOUT bit state has changed, record the OVPCOF value as V_{OS2} and then go to Step 6.

- Step 6

  Restore the comparator input offset voltage calibration value V_{OS} into the OVPCOF[4:0] bit field. The offset calibration procedure is now finished.

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

  Residue $V_{OS}=V_{OUT}-V_{IN}$.

Analog to Digital Converter – ADC

The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the microcontroller, the need for external components is reduced significantly with the corresponding follow-on benefits of lower costs and reduced component space requirements.

A/D Converter Overview

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals (such as that from sensors or other control signals) and convert these signals directly into a 12-bit digital value. It also can convert the internal signalsThe external or internal analog signal to be converted is determined by the SAINS3~SAINS0 and SACS3~SACS0 bits. Note that when the internal analog signal is selected to be converted, the external channel analog input will be automatically switched off. More detailed information about the A/D converter input signal is described in the "A/D Converter Control Registers" and "A/D Converter Input Signals" sections respectively.

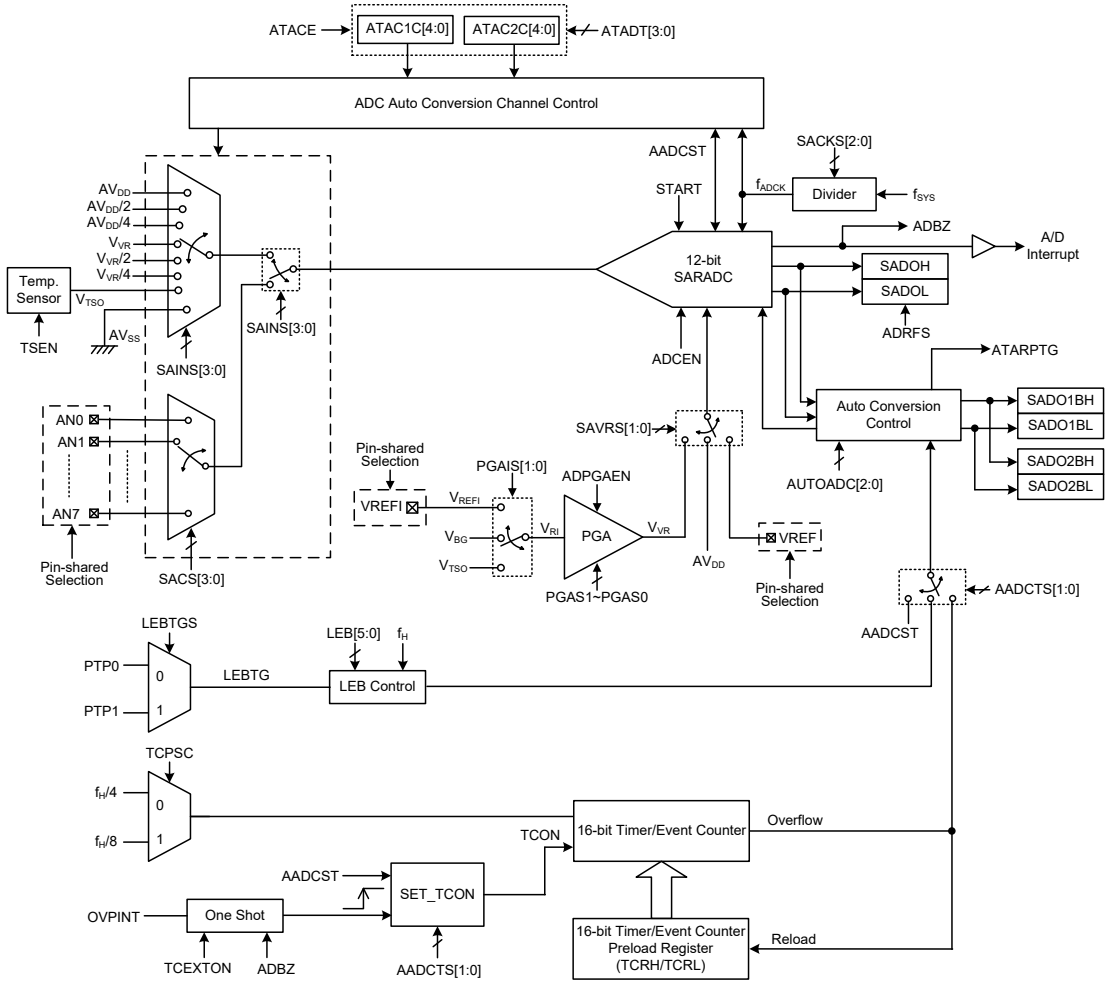
External Input Channels	Internal Input Signals
8: AN0~AN7	AV_{DD}, $AV_{DD}/2$, $AV_{DD}/4$, V_{VR}, $V_{VR}/2$, $V_{VR}/4$, V_{TSO}, AV_{SS}

The A/D converter can be operated in manual trigger conversion mode, 1-channel automatic conversion mode, and 2-channel automatic conversion mode. The manual trigger conversion is to start an A/D conversion by controlling the START bit using the user program, which is the general A/D converter function. The automatic trigger conversion is to start a conversion automatically when receiving an active trigger signal. The trigger signal is selected by the AADCTS1~AADCTS0 bits in the SADC4 register. The number of conversions in a group of automatic conversion is defined by the AUTOADC2~AUTOADC0 bits in the same register, which can be 1, 2, 4, 8 or 16.

AUTOADC[2:0] Bits	ATACE Bit	A/D Conversion Mode	Channel Input Select Bits
000	x	Manual trigger an A/D conversion	SAINS[3:0] & SACS[3:0]
001~111	0	1-channel automatic conversion	SAINS[3:0] & SACS[3:0]
001~111	1	2-channel automatic conversion	CH1: ATAC1C[4:0] CH2: ATAC2C[4:0]

"x": Don't care

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



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using several registers. A read only register pair exists to store the A/D converter data 12-bit value. Two register pairs, SADO1BH/SADO1BL and SADO2BH/SADO2BL, are used to store the cumulative automatic conversion result of the CH1 and CH2 in the automatic A/D conversion mode. The remaining registers are control registers which setup the operating and control functions of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADO1BH	D15	D14	D13	D12	D11	D10	D9	D8
SADO1BL	D7	D6	D5	D4	D3	D2	D1	D0

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADO2BH	D15	D14	D13	D12	D11	D10	D9	D8
SADO2BL	D7	D6	D5	D4	D3	D2	D1	D0
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
SADC2	ADPGAEN	PGAIS1	—	PGAIS0	SAVRS1	SAVRS0	PGAGS1	PGAGS0
SADC3	—	—	—	—	—	—	—	TSEN
SADC4	AADCTS1	AADCTS0	—	ATARPTG	—	AUTOADC2	AUTOADC1	AUTOADC0
LEBC	AADCST	LEBTGS	LEB5	LEB4	LEB3	LEB2	LEB1	LEB0
ATAC1C	—	—	ATACE	ATAC1SS	ATAC1S3	ATAC1S2	ATAC1S1	ATAC1S0
ATAC2C	—	—	—	ATAC2SS	ATAC2S3	ATAC2S2	ATAC2S1	ATAC2S0
ATADT	—	—	—	—	ATADT3	ATADT2	ATADT1	ATADT0
TCRC	—	—	—	TCON	—	—	TCEXTON	TCPSC
TCRL	D7	D6	D5	D4	D3	D2	D1	D0
TCRH	D15	D14	D13	D12	D11	D10	D9	D8

A/D Converter Register List

A/D Converter Data Registers

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 the A/D converter data register contents will remain unchanged if the A/D converter is disabled.

If the A/D converter operates in the automatic conversion mode (AUTOADC[2:0]≠000), the ADRFS bit will be fixed at "1" by the hardware. The conversion result data formart is: D[11:8]=SADOH[3:0], D[7:0]=SADOL[7:0], the unused bits in the SADOH register are read as zero.

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	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D Converter Data Registers

The device contains two pairs of registers which are the data buffer registers to store the CH1 and CH2 cumulative conversion result when the A/D converter operates in the automatic conversion mode.

Register	SADO1BH								SADO1BL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

A/D Automatic Conversion Data Buffer 1 Registers

Register	SADO2BH								SADO2BL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

A/D Automatic Conversion Data Buffer 2 Registers

A/D Converter Control Registers

To control the function and operation of the A/D converter, several control registers 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 converter operating mode, the A/D converter clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SAINS3~ SAINS0 bits in the SADC1 register and SACS3~SACS0 bits in the SADC0 register are used to select which analog signal from either the external or internal signals will be connected to the A/D converter when the A/D converter is in the manual trigger conversion mode or the 1-channel automatic conversion mode. If the A/D converter operates in the 2-channel automatic conversion mode, the CH1 and CH2 input signals are selected by the ATAC1C and ATAC2C registers respectively. The LEBC register is used to enable the automatic conversion, select the LEBTG trigger signal source and set the trigger signal leading edge blanking time. The SADC4 register is used to select the automatic conversion start trigger signal and the number of conversions. The ATAC1C, ATAC2C and ATADT registers can be used to enable the 2-channel automatic conversion mode, select the CH1 and CH2 signals and the time interval between two conversions.

The A/D converter also contains a programmable gain amplifier, PGA, to generate the A/D converter internal reference voltage. The overall operation of the PGA is controlled using the SADC2 register. The SADC3 register is used to enable/disable the integrated temperature sensor circuitry.

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 converter 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 (for manual trigger conversion)
0→1→0: Start A/D conversion
Note: This bit is valid only when the AUTOADC[2:0] is “000”.
This bit is used to initiate an A/D conversion process in the manual trigger conversion mode. 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 a certain condition as listed in the table occurs, the ADBZ flag will be set high

to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to zero after the A/D conversion is complete.

AUTOADC[2:0]	AADCTS[1:0]	ADBZ Set High Condition
=000 (manual trigger conversion mode)	xx	START bit is set from low to high and then low again
≠000 (automatic trigger conversion mode)	00	LEB falling edge
	01/10	Timer counter overflows
	11	AADCST bit is set from low to high

Bit 5 **ADCEN**: A/D Converter function enable control
0: Disable
1: Enable

This bit controls the A/D converter 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 converter data register pair, SADOH and SADOL, will remain unchanged.

Bit 4 **ADRFS**: A/D Converter data format control
0: ADC output data format → SADOH=D[11:4]; SADOL=D[3:0]
1: ADC output data format → SADOH=D[11:8]; SADOL=D[7:0]

This bit controls the format of the 12-bit converted A/D converter value in the two A/D converter data registers.

Note: This bit can be changed by software when the AUTOADC[2:0] is "000". When the AUTOADC[2:0] bits are not "000", the ADRFS bit is fixed at 1 by the hardware.

Bit 3~0 **SACS3~SACS0**: Manual or 1-CH automatic conversion mode external analog input channel selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000~1111: Undefined, input floating

These bits are used to select which external analog input channel is to be converted in the manual trigger conversion or automatic trigger 1-CH conversion mode.

• SADC1 Register

Bit	7	6	5	4	3	2	1	0
Name	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	0	0	0	0	—	0	0	0

Bit 7~4 **SAINS3~SAINS0**: Manual or 1-CH automatic conversion mode input signal selection
0000: External source – External analog channel input, ANn
0001: Internal source – Internal A/D converter power supply voltage AV_{DD}
0010: Internal source – Internal A/D converter power supply voltage $AV_{DD}/2$
0011: Internal source – Internal A/D converter power supply voltage $AV_{DD}/4$
0100: External source – External analog channel input, ANn
0101: Internal source – Internal A/D converter PGA output voltage V_{VR}
0110: Internal source – Internal A/D converter PGA output voltage $V_{VR}/2$
0111: Internal source – Internal A/D converter PGA output voltage $V_{VR}/4$
1000~1010: Internal source – Ground, AV_{SS}
1011: Internal source – Internal A/D converter temperature sensor output, V_{TSO}
1100~1111: External source – External analog channel input, ANn

These bits are available in the manual trigger conversion or automatic trigger 1-CH conversion mode.

When the internal analog signal is selected to be converted, the external channel input signal will automatically be switched off regardless of the SACS3~SACS0 bit value. This will prevent the external channel input from being connected together with the internal analog signal.

Bit 3 Unimplemented, read as "0"

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source (f_{ADCK}) selection
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: $f_{SYS}/128$

• SADC2 Register

Bit	7	6	5	4	3	2	1	0
Name	ADPGAEN	PGAIS1	—	PGAIS0	SAVRS1	SAVRS0	PGAGS1	PGAGS0
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 **ADPGAEN**: A/D converter PGA enable/disable control
0: Disable
1: Enable

This bit is used to control the A/D converter internal PGA function. When the PGA output voltage is selected as A/D input or A/D reference voltage, the PGA needs to be enabled by setting this bit high. Otherwise the PGA needs to be disabled by clearing the ADPGAEN bit to zero to conserve power.

Bit 6,4 **PGAIS1~PGAIS0**: PGA input voltage (V_{RI}) selection
00: External VREFI pin
01: Internal reference voltage V_{BG} (0.93V)
1x: Internal temperature sensor output voltage V_{TSO}

When the internal reference voltage V_{BG} or the temperature sensor output voltage V_{TSO} is selected as the PGA input, the external reference voltage on the VREFI pin will be automatically switched off. In addition, the internal reference V_{BG}, if selected, should be enabled by setting the VBGEN bit in the LVDC register to "1".

Bit 5 Unimplemented, read as "0"

Bit 3~2 **SAVRS1~SAVRS0**: A/D converter reference voltage selection
00: Internal A/D converter power, AV_{DD}
01: External VREF pin
1x: Internal PGA output voltage, V_{VR}

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

Note that when the A/D input signal comes from the temperature sensor output V_{TSO}, it is recommended to set these bits to select the V_{VR} reference voltage.

Bit 1~0 **PGAGS1~PGAGS0**: PGA gain selection
00: Gain=1
01: Gain=1.720 – V_{VR}=1.6V if V_{RI}=0.93V
10: Gain=3.226 – V_{VR}=3.0V if V_{RI}=0.93V
11: Gain=4.301 – V_{VR}=4.0V if V_{RI}=0.93V

Note that when the A/D input signal comes from the temperature sensor output V_{TSO}, it is recommended to set these bits to select a V_{VR} voltage of 1.6V.

• **SADC3 Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1 **D1**: Reserved bit, keep unchanged after power-on

Bit 0 **TSEN**: Temperature sensor enable/disable 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.

• **SADC4 Register**

Bit	7	6	5	4	3	2	1	0
Name	AADCTS1	AADCTS0	—	ATARPTG	—	AUTOADC2	AUTOADC1	AUTOADC0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

Bit 7~6 **AADCTS1~AADCTS0**: Automatic A/D conversion trigger selection
00: LEBTG rising edge
01: Timer/Event Counter overflow – TCON is set high by an OVPINT rising edge
10: Timer/Event Counter overflow – TCON is set high by the AADCST bit changing from 0 to 1
11: Software bit trigger – change AADCST from 0 to 1 to start a conversion

Bit 5 Unimplemented, read as "0"

Bit 4 **ATARPTG**: Automatic A/D conversion mode signal retrigger flag
0: No trigger signal occurs in A/D conversion
1: A trigger signal occurs in A/D conversion

When the AADCST bit is changed from 0 to 1 to enable the automatic conversion mode, the hardware automatically clears the ATARPTG bit to zero. This bit is only set high by hardware when a retrigger occurs during an A/D conversion (ADBZ=1). It will not be set high by hardware within the ATADT[3:0] defined delay time. After being set high, it can be cleared to zero by software.

Bit 3 Unimplemented, read as "0"

Bit 2~0 **AUTOADC2~AUTOADC0**: Selection for the number of automatic conversions
000: N=0 – automatic conversion function is disabled
001: N=1
010: N=2
011: N=4
100: N=8
101~111: N=16

These three bits define the number of conversions, N, in a group of automatic conversions. When AUTOADC[2:0]=000 (i.e. N=0), the automatic conversion function is disabled.

If N≠0 and ATACE=0, after the A/D converter performs N times of conversions, the cumulative automatic conversion result is stored in the SADO1BH and SADO1BL register pair and the ADF flag is set high. If N≠0 and ATACE=1, after the A/D converter performs N times of conversions, the cumulative automatic conversion result of the CH1 is stored in the SADO1BH and SADO1BL register pair and the CH2 is stored in the SADO2BH and SADO2BL register pair and the ADF flag is set high.

Note: 1. When AUTOADC[2:0]≠000, the START bit in the SADC0 register is invalid. A group of automatic conversions is enabled by setting the AADCST bit high. When the selected conversion start trigger signal occurs, an A/D conversion is started automatically.

2. In the automatic conversion mode, after an conversion, the A/D converter output data format is: D[11:8]=SADOH[3:0], D[7:0]=SADOL[7:0]; the unused bits in SADOH are all zero.

• LEBC Register

Bit	7	6	5	4	3	2	1	0
Name	AADCST	LEBTGS	LEB5	LEB4	LEB3	LEB2	LEB1	LEB0
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 **AADCST**: Automatic A/D conversion control
0: Disable
1: Enable
The AADCST bit is set high by software. This bit can be cleared to zero by hardware or software. This bit is cleared to zero after N (defined by AUTOADC[2:0]) automatic conversions are completed. When the automatic conversion trigger source suddenly stops or an exception occurs, this bit can be cleared to zero by software. For related applications, refer to the A/D Automatic Conversion Software Stop Descriptions section. When this bit is set high, the ATARPTG bit is automatically cleared.
Note: This bit is invalid when the AUTOADC[2:0] bit field is "000".

Bit 6 **LEBTGS**: LEBTG source selection
0: From PTM0 output PTP0
1: From PTM1 output PTP1

Bit 5~0 **LEB5~LEB0**: LEB (Leading-Edge Blanking) time control
LEB time=(LEB[5:0]+1)×8×1/f_H, LEB[5:0]=0~63
For example: f_H=16MHz
LEB[5:0]=0, LEB time=(0+1)×8×0.0625μs=0.5μs
LEB[5:0]=5, LEB time=(5+1)×8×0.0625μs=3μs
Note: The LEB[5:0] is only valid when the LEBTG rising edge trigger is selected (AADCTS[1:0]=00).

• ATAC1C Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	ATACE	ATAC1SS	ATAC1S3	ATAC1S2	ATAC1S1	ATAC1S0
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 **ATACE**: Automatic Trigger ADC 2-channel conversion function control
0: Disable
1: Enable
When ATACE=1, the automatic conversion starts when the AADCST bit is changed from low to high and a trigger source determined by the AADCTS[1:0] bit filed is active. In the automatic conversion mode the START bit is invalid.

Bit 4 **ATAC1SS**: Internal or external input source selection for Automatic Trigger ADC input channel 1
0: From internal signal
1: From external analog channel input, ANn
If ATACE=0, the ATAC1SS bit is invalid.

Bit 3~0 **ATAC1S3~ATAC1S0**: Input signal selection for Automatic Trigger ADC input channel 1

If ATAC1SS=0:

0000: A/D converter power supply voltage AV_{DD}
0001: A/D converter power supply voltage $AV_{DD}/2$
0010: A/D converter power supply voltage $AV_{DD}/4$
0011: A/D converter PGA output voltage V_{VR}
0100: A/D converter PGA output voltage $V_{VR}/2$
0101: A/D converter PGA output voltage $V_{VR}/4$
0110~1000: Ground, AV_{SS}
1001: Temperature sensor output, V_{TSO}
Others: A/D converter power supply voltage AV_{DD}

If ATAC1SS=1:

0000: AN0
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000~1111: Undefined, input floating

If ATACE=0, the ATAC1S[3:0] bits are invalid.

- **ATAC2C Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	ATAC2SS	ATAC2S3	ATAC2S2	ATAC2S1	ATAC2S0
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 **ATAC2SS**: Internal or external input source selection for Automatic Trigger ADC input channel 2

0: From internal signal
1: From external analog channel input, ANn

If ATACE=0, the ATAC2SS bit is invalid.

Bit 3~0 **ATAC2S3~ATAC2S0**: Input signal selection for Automatic Trigger ADC input channel 2

If ATAC2SS=0:

0000: A/D converter power supply voltage AV_{DD}
0001: A/D converter power supply voltage $AV_{DD}/2$
0010: A/D converter power supply voltage $AV_{DD}/4$
0011: A/D converter PGA output voltage V_{VR}
0100: A/D converter PGA output voltage $V_{VR}/2$
0101: A/D converter PGA output voltage $V_{VR}/4$
0110~1000: Ground, AV_{SS}
1001: Temperature sensor output, V_{TSO}
Others: A/D converter power supply voltage AV_{DD}

If ATAC2SS=1:

0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6

0111: AN7
1000~1111: Undefined, input floating
If ATACE=0, the ATAC2S[3:0] bits are invalid.

• **ATADT Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	ATADT3	ATADT2	ATADT1	ATADT0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **ATADT3~ATADT0**: Automatic A/D conversion delay time setting
0000: $1\times t_{ADCK}$
0001: $2\times t_{ADCK}$
0010: $3\times t_{ADCK}$
…
…
1110: $15\times t_{ADCK}$
1111: $16\times t_{ADCK}$
Note: $t_{ADCK}=1/f_{ADCK}$
In the automatic trigger conversion mode (AUTOADC[2:0]≠000), these four bits define the delay time between two conversions when ATACE=0 & AADCTS[1:0]=11, or when ATACE=1.

Timer/Event Counter Registers

A 16-bit timer is included in the A/D converter circuitry to carry out timing function and its overflow event can be used as the A/D automatic conversion start trigger signal. There are three registers related to the timer/event counter, including a control register, TCRC, and a pair of preload registers, TCRH and TCRL. The TCRC register defines the counter enabling method and selects its counting clock. When the TCON bit is set high, the timer will count from the initial value loaded by the preload register to the full count of FFFFH, at which point the timer overflows.

When AADCST=1, the software should avoid writing values to the TCRC register to prevent malfunction.

• **TCRC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	TCON	—	—	TCEXTON	TCPSC
R/W	—	—	—	R/W	—	—	R/W	R/W
POR	—	—	—	0	—	—	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4 **TCON**: Timer/event counter counting enable
0: Disable
1: Enable
This bit can only be cleared to zero by software but cannot be set high by the software.
When AADCTS[1:0]=01: The TCON bit is set to 1 automatically at a rising edge of the OVPINT signal and then the counter starts counting, and will stop counting when it is full and overflows, which will clear the TCON bit.
When AADCTS[1:0]=10: The TCON bit is set to 1 automatically when the AADCST bit changes from 0 to 1 and then the counter starts counting. When the AADCST bit changes from 1 to 0, the TCON bit will be cleared to zero automatically.
Note: The TCR counter is written with a new value when TCON=0 or it is reloaded only when TCON=1 and the counter overflows.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **TCEXTON**: Enable control for external rising edge triggerring the TCON
0: Disable
1: Enable
When this bit is 0, the OVPINT signal rising edges have no effect on the TCON bit. When this bit is 1 and the ADBZ bit is 0, the TCON bit will be set high when an OVPINT rising edge is received.

Bit 0 **TCPSC**: Timer prescaler rate selection
0: $f_H/4$
1: $f_H/8$

- **TCRH 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**: Timer preload register high byte

- **TCRL 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**: Timer preload register low byte

Note that writing to the TCRL register will only write the data into an internal lower byte buffer while writing to the TCRH register will transfer the high byte data and the contents of the lower byte buffer into the TCRH and TCRL registers respectively. Therefore the timer counter preload register is changed by each write operation to the TCRH register. Reading from the TCRH register will latch the contents of the TCRH and TCRL counters to the destination and the lower byte buffer respectively, while reading from TCRL will read the contents of the lower byte buffer.

A/D Converter Reference Voltage

The actual reference voltage supply to the A/D Converter can be supplied from the positive power supply, AV_{DD}, an external reference source supplied on pin VREF, or an internal reference source derived from the PGA output voltage, V_{VR}. The desired selection is made using the SAVRS1~SAVRS0 bits in the SADC2 register. The internal reference voltage is amplified through a programmable gain amplifier, PGA, which is controlled by the ADPGAEN bit in the SADC2 register. The PGA gain is selected using the PGAGS1~PGAGS0 bits in the SADC2 register. The PGA input can come from the external reference input pin, VREFI, or an internal Bandgap reference voltage, V_{BG}, or the internal temperature sensor output voltage, V_{TSO}, selected by the PGAIS1~PGAIS0 bits in the SADC2 register. The internal Bandgap reference circuit or the temperature sensor should first be enabled before the V_{BG} or V_{TSO} is selected to be used.

As the VREFI and VREF pins are pin-shared with other functions, when the VREFI or VREF pin is selected as the reference voltage input pin, the VREFI or VREF pin-shared function selection bits should first be properly configured to disable other pin-shared functions. However, if the internal reference signal is selected as the reference source, the external reference input from the VREFI or VREF pin will automatically be switched off by hardware.

Note that the analog input signal values must not be allowed to exceed the value of the selected A/D Converter reference voltage.

SAVRS[1:0]	Reference Source	Description
00	AV_{DD}	Internal A/D converter power supply voltage
01	VREF pin	External A/D converter reference pin
10 or 11	V_{VR}	Internal A/D converter PGA output voltage

A/D Converter Reference Voltage Selection

A/D Converter Input Signals

All of the external A/D Converter analog input pins are pin-shared with the I/O pins as well as other functions. The corresponding pin-shared function selection bits in the PxSn registers, determine whether the external input pins are setup as A/D converter analog channel inputs or whether they have other functions. If the corresponding pin is setup to be an A/D converter analog channel input, the original pin 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 Converter input as when the relevant A/D converter input function selection bits enable an A/D converter input, the status of the port control register will be overridden.

The A/D converter also has eight internal anlog signals, the A/D converter positive power supply voltage, AV_{DD}, and its subdivided versions, $AV_{DD}/2$ and $AV_{DD}/4$; the PGA output voltage, V_{VR}, and its subdivided versions, $V_{VR}/2$ and $V_{VR}/4$; the temperature sensor output, V_{TSO}; or the A/D converter negative power supply, AV_{SS}. As this device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter.

For manual trigger A/D conversion mode and the 1-CH A/D automatic conversion mode, the SAINS3~SAINS0 bits in the SADC1 register are used to determine whether the analog signal to be converted comes from an external channel input or an internal analog signal. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted.

For the 2-CH A/D automatic conversion mode, the input signals of the CH1 and CH2 are selected by the ATAC1C and ATAC2C registers. The ATACnSS bit is used to determine whether the analog signal to be converted comes from the external channel input or internal analog signal. The ATACnS3~ATACnS0 bits are used to determine which external channel input or which internal signal is selected to be converted.

If the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off. It will prevent the external channel input from being connected together with the internal analog signal.

SAINS[3:0]	SACS[3:0]	Input Signals	Description
0000, 0100, 1100~1111	0000~0111	AN0~AN7	External pin analog input
	1000~1111	—	Non-existed channel, input is floating
0001	xxxx	AV_{DD}	A/D converter power supply voltage
0010	xxxx	$AV_{DD}/2$	A/D converter power supply voltage divided by 2
0011	xxxx	$AV_{DD}/4$	A/D converter power supply voltage divided by 4
0101	xxxx	V_{VR}	A/D converter PGA output voltage
0110	xxxx	$V_{VR}/2$	A/D converter PGA output voltage divided by 2
0111	xxxx	$V_{VR}/4$	A/D converter PGA output voltage divided by 4
1000~1010	xxxx	AV_{SS}	A/D converter negative power supply
1011	xxxx	V_{TSO}	A/D converter temperature sensor output voltage

A/D Converter Input Signal Selection – Manual or 1-CH Auto Mode

ATAC1SS/ ATAC2SS	ATAC1S[3:0]/ ATAC2S[3:0]	CH1/CH2 Input Signals	Description
0	0000, 1010~1111	AV_{DD}	A/D converter power supply voltage
	0001	$AV_{DD}/2$	A/D converter power supply voltage divided by 2
	0010	$AV_{DD}/4$	A/D converter power supply voltage divided by 4
	0011	V_{VR}	A/D converter PGA output voltage
	0100	$V_{VR}/2$	A/D converter PGA output voltage divided by 2
	0101	$V_{VR}/4$	A/D converter PGA output voltage divided by 4
	0110~1000	AV_{SS}	A/D converter negative power supply
	1001	V_{TSO}	A/D converter temperature sensor output voltage
1	0000~0111	AN0~AN7	External pin analog input
	1000~1111	—	Non-existed channel, input is floating

A/D Converter Input Signal Selection – 2-CH Auto Mode

A/D Converter Clock Source

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 conversion clock source is determined by the system clock f_{SYS}, and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D conversion clock source speed that can be selected. As the recommended value of permissible A/D conversion clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken when selecting the clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the SACKS2~SACKS0 bits should not be set to "000", "110" or "111". Doing so will give A/D conversion clock periods that are less than the minimum A/D conversion clock period or greater than the maximum A/D conversion clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * special care must be taken.

Note that if the input signal to be converted is the temperature sensor output voltage, the permissible A/D conversion clock period is 4μs to 10μs.

f_{SYS}	A/D Conversion 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

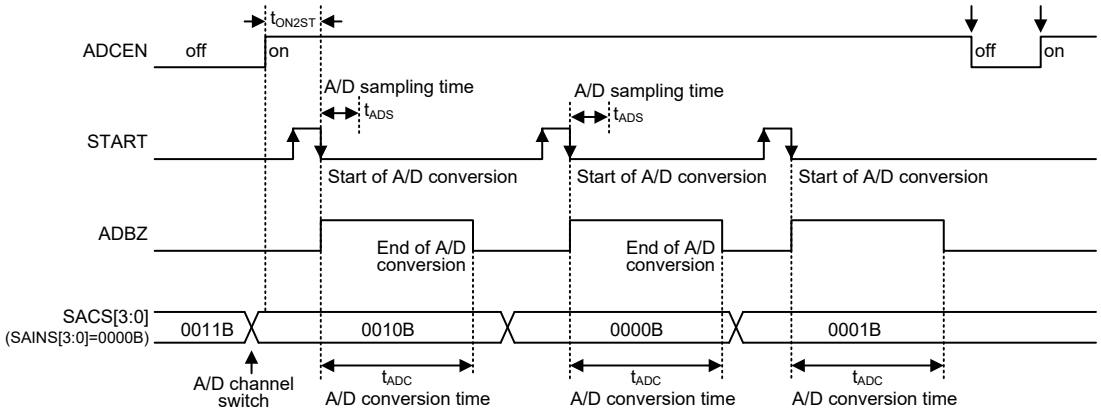
A/D Conversion Clock Period Examples

A/D Conversion Rate and Timing Diagram

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

Maximum single A/D conversion rate=1/(A/D conversion clock period×16)

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



A/D Conversion Timing – Manual Trigger Mode & External Channel Input

Manual Trigger A/D Conversion

This is the general A/D converter operating mode (AUTOADC[2:0]=000). The start of each A/D conversion is triggered manually. The A/D converter will perform conversions on the channel specified by the the SAINS[3:0] bits in the SADC1 register together with the SACS[3:0] bits in the SADC0 register.

The START bit in the SADC0 register is used to start the A/D 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 has been successfully initiated. When an A/D conversion is complete, the ADBZ will be cleared to "0". In addition, the corresponding A/D converter 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 converter internal interrupt signal will direct the program flow to the associated A/D converter 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 following summarises the individual steps that should be executed in order to implement a manual trigger A/D conversion process.

- Step 1

  Select the required A/D conversion clock by properly programming the SACKS[2:0] bits in the SADC1 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 SAINS[3:0] bits in the SADC1 register.

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 is selected to come from the external channel analog signal by configuring the SAINS[3:0] bits, the desired external channel input is selected by configuring the SACS[3:0] bits. When the A/D input signal comes from the external channel input, the corresponding pin should be configured as an A/D input function in advance by configuring the relevant pin-shared function control bits. Then go to Step 6.

- Step 5

  If the A/D input signal is selected to come from the internal analog signal by configuring the SAINS[3:0] bits, the external channel analog signal input will be automatically switched off regardless of the SACS[3:0] bit value. After this step, go to Step 6.

- Step 6

  Select the reference voltage source by configuring the SAVRS[1:0] bits in the SADC1 register.

- Step 7

  Select the A/D converter output data format in advance by configuring the ADRFS bit in the SADC0 register.

- Step 8

  If the A/D converter interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, the A/D converter interrupt control bit, ADE, and its corresponding multi-function interrupt control bit must all be set high.

- Step 9

  The A/D conversion procedure can now be initialised by setting the START bit from low to high and then low again.

- Step 10

  If an A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process has completed, the ADBZ flag will go low after which the output data can be read from the 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.

Automatic Trigger 1-Channel A/D Conversion

In the 1-channel automatic A/D conversion mode (ATACE=0, AUTOADC[2:0]≠000), the A/D converter will perform conversions on the channel specified by the SAINS[3:0] bits in the SADC1 register together with the SACS[3:0] bits in the SADC0 register. After configuring basic A/D converter parameters, users should select the start trigger signal and setup the number of automatic conversions. The LEB time (LEB[5:0]) should be configured if the trigger signal is selected to be the LEBTG rising edge, the timer counter related parameters should be configured if the trigger signal is timer counter overflow, and the delay time between two conversions (ATADT[3:0]) should be configured if the trigger signal is the AADCST bit changing from 0 to 1.

When the AADCST bit is changed from 0 to 1, the automatic conversion is enabled. When an active trigger signal source occurs, it may activate the LEB timing or the timer counter to start counting or start an A/D conversion immediately. The converted data will be stored in the SADO1BH and SADO1BL registers from the SADOH and SADOL registers. After the whole group of automatic conversions is completed, the value in SADO1B register is the cumulative conversion result of these

conversions.

The following summarises the individual steps that should be executed in order to implement an automatic trigger 1-channel A/D conversion process.

- Step 1

  Select the required A/D conversion clock by properly programming the SACKS[2:0] bits in the SADC1 register.

- Step 2

  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to "1".

- Step 3

  Enable the 1-channel automatic conversion function by clearing the ATACE bit in the ATAC1C register to "0".

- Step 4

  Select which signal is to be connected to the internal A/D converter in 1-channel automatic conversion mode by correctly configuring the SAINS[3:0] bits in the SADC1 register.

  Select the external channel input to be converted, go to Step 5.

  Select the internal analog signal to be converted, go to Step 6.

- Step 5

  If the A/D input signal is selected to come from the external channel analog signal by configuring the SAINS[3:0] bits, the desired external channel input is selected by configuring the SACS[3:0] bits. When the A/D input signal comes from the external channel input, the corresponding pin should be configured as an A/D input function by configuring the relevant pin-shared function control bits. Then go to Step 7.

- Step 6

  If the A/D input signal is selected to come from the internal analog signal by configuring the SAINS[3:0] bits, the external channel analog input will automatically be switched off regardless of the SACS[3:0] bit value. Then go to Step 7.

- Step 7

  Select the number of automatic A/D conversions by setting the AUTOADC[2:0] bits in the SADC4 register.

- Step 8

  Set the trigger signal of the ADC automatic conversion via the AADCTS[1:0] bits in the SADC4 register.

  If the LEBTG rising edge is selected (00B), go to Step 9.

  If the OVPINT rising edge activated timer overflow event is selected (01B), go to Step 10.

  If the AADCST rising edge activated timer overflow event is selected (10B), go to Step 11.

  If the AADCST rising edge immediate trigger is selected (11B), go to Step 13.

- Step 9

  Set the blanking time before the A/D conversion starts by configuring the LEB[5:0] bits in the LEBC register, then go to Step 14.

- Step 10

  Set the TCEXTON bit in the TCRC register to enable the OVPINT rising edge to trigger the timer, then go to Step 11.

- Step 11

Set the timer clock source through the TCPSC bit in the TCRC register, go to Step 12.

- Step 12

  Set the timer preload value using the TCRH and TCRL registers, go to Step 14.

- Step 13

  Set the delay time between data conversions using the ATADT[3:0] bits in ATADT register, go to Step 14.

- Step 14

  Select the required reference voltage using the SAVRS[1:0] bits in the SADC1 register.

- Step15

  If the A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, the A/D converter interrupt control bit, ADE, and its corresponding multi-function interrupt control bit must all be set high.

- Step 16

  Set the AADCST bit in the LEBC register from low to high, the A/D converter will start the conversion process when the selected trigger signal occurs.

Note: 1. After the A/D automatic conversion is completed, the AADCST bit will be cleared by the hardware and the A/D converter interrupt flag ADF bit will be set high.

2. When checking for the end of the automatic conversion process, if the method of polling the AADCST bit in the LEBC register is used, the interrupt enable step above can be omitted.

Automatic Trigger 2-Channel A/D Conversion

In the 2-channel automatic A/D conversion mode (ATACE=1, AUTOADC[2:0]≠000), the A/D converter CH1 and CH2 input signal is selected by using the ATACnSS and ATACnS[3:0] bits respectively. After the basic A/D converter parameters are set, set the ATACE bit high to enable the 2-channel automatic conversion mode, select the number of automatic conversions and the trigger signal in automatic mode, and set the interval time between two conversions. The LEB time (LEB[5:0]) should be configured if the trigger signal is selected to be the LEBTG rising edge, and the timer counter related parameters should be configured if the trigger signal is the timer counter overflow.

When the AADCST bit is changed from 0 to 1, the automatic conversion is enabled. When an active trigger signal source occurs, it may activate the LEB timing or the timer counter to start counting or start an A/D conversion immediately. The A/D converter performs the CH1 conversion first. When the specified interval time has elapsed, the A/D converter will perform the CH2 conversion. The CH1 converted data will be stored in the SADO1BH and SADO1BL registers from the SADOH and SADOL registers and the CH2 converted data will be stored in the SADO2BH and SADO2BL registers from the SADOH and SADOL registers. After the specified number of A/D automatic conversions, the 2-channel automatic conversion is completed. The value in the SADO1B register is the cumulative result of CH1 conversions and the value in the in the SADO2B register is the cumulative result of CH2 conversions.

In the 2-channel automatic A/D conversion mode, the two channels will use the same A/D reference voltage. If either of the two channel inputs comes from the temperature sensor, the reference voltage and A/D conversion period t_{ADCK} must be configured based on the temperature sensor requirement, which requires to use the V_{VR} reference voltage of 1.6V and set the t_{ADCK} in a range of 4μs~10μs.

The following summarises the individual steps that should be executed in order to implement an

automatic trigger 2-channel A/D conversion process.

- Step 1

  Select the required A/D conversion clock by properly programming the SACKS[2:0] bits in the SADC1 register.

- Step 2

  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to "1".

- Step 3

  Enable the 2-channel automatic conversion function by setting the ATACE bit in the ATAC1C register to "1".

- Step 4

  Select whether an external or internal signal is to be connected to the Automatic Trigger ADC input channel 1 and Automatic Trigger ADC input channel 2 by correctly configuring the ATACnSS bit in the ATACnC register respectively.

  Select the external channel input to be converted (ATACnSS=1), go to Step 5.

  Select the internal analog signal to be converted (ATACnSS=0), go to Step 6.

- Step 5

  The desired external channel input can be selected by configuring the ATACnS[3:0] bits. As the external channel input pin is pin-shared with other functions, the corresponding pin should be configured as an A/D input function in advance by configuring the relevant pin-shared function control bits. Then go to Step 7.

- Step 6

  The desired internal analog signal can be selected by configuring the ATACnS[3:0] bits. Then go to Step 7.

- Step 7

  Select the number of automatic A/D conversions by setting the AUTOADC[2:0] bits in the SADC2 register.

- Step 8

  Set the trigger signal of the ADC automatic conversion via the AADCTS[1:0] bits in the SADC2 register.

  If the LEBTG rising edge is selected (00B), go to Step 9.

  If the OVPINT rising edge activated timer overflow event is selected (01B), go to Step 10.

  If the AADCST rising edge activated timer overflow event is selected (10B), go to Step 11.

  If the AADCST rising edge immediate trigger is selected (11B), go to Step 13.

- Step 9

  Set the blanking time before the A/D conversion starts by configuring the LEB[5:0] bits in the LEBC register, then go to Step 13.

- Step 10

  Set the TCEXTON bit in the TCRC register to enable the OVPINT rising edge to trigger the timer, go to Step 11.

- Step 11

  Set the timer clock source through the TCPSC bit in the TCRC register, go to Step 12.

- Step 12

Set the timer preload value using the TCRH and TCRL registers, go to Step 13.

- Step 13

  Setup the delay time between two conversions by configuring the ATADT[3:0] bits in the ATADT register.

  If AADCTS[1:0]=00B/01B/10B, this delay time is inserted only between CH1 conversion end to CH2 conversion start. If AADCTS[1:0]=11B, this delay time is inserted between one conversion end to the next converstion start (CH1→CH2→CH1…).

- Step 14

  Select the required reference voltage using the SAVRS[1:0] bits in the SADC1 register.

- Step 15

  If the A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, the A/D converter interrupt control bit, ADE, and its corresponding multi-function interrupt control bit must all be set high.

- Step 16

  Set the AADCST bit in the LEBC register from low to high, the A/D converter will start the conversion process when the selected trigger signal occurs.

Note: 1. After the A/D automatic conversion is completed, the AADCST bit will be cleared by the hardware and the A/D converter interrupt flag ADF bit will be set high.
2. When checking for the end of the automatic conversion process, if the method of polling the AADCST bit in the LEBC register is used, the interrupt enable step above can be omitted.

Automatic Trigger Conversion Description

An automatic conversion can be initiated by an LEBTG rising edge with an LEB time, an overflow event of the 16-bit timer counter which is started by an OVPINT signal rising edge or by setting the AADCST bit, or initiated directly by setting the AADCST bit from low to high, which is selected by the AADCTS[1:0] bits. The following section describes how the A/D converter is started by these triggers and completes a group of automatic conversions.

AADCTS[1:0]=00B

When the AADCTS[1:0] bits are 00 and the AUTOADC[2:0] bits are not equal to 000, setting the AADCST bit high will enable the A/D automatic synchronization delay conversion, which is triggered by an LEBTG rising edge. Here the actual LEBTG source can be PTP0 output from PTM1 or the PTP1 output from PTM1, which is selected by the LEGTGS bit in the LEBC register. After setting the AADCST bit high, each LEBTG rising edge will start the leading edge blanking circuit. After the LEB time configured by the LEB[5:0] bits, an A/D conversion will start. After N automatic conversions have been completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set high. Here the number N is defined by the AUTOADC[2:0] bits. If the LEBTG source stops abnormally, the AADCST bit can be cleared to zero by the software.

The AADCST bit is unable to provide a repeatable trigger. Once the AADCST bit is set high, the circuit cannot be triggered until N automatic conversions are completed. In program design it should be noted that after the AADCST bit is cleared, the ADF flag should also be cleared using the application progrm.

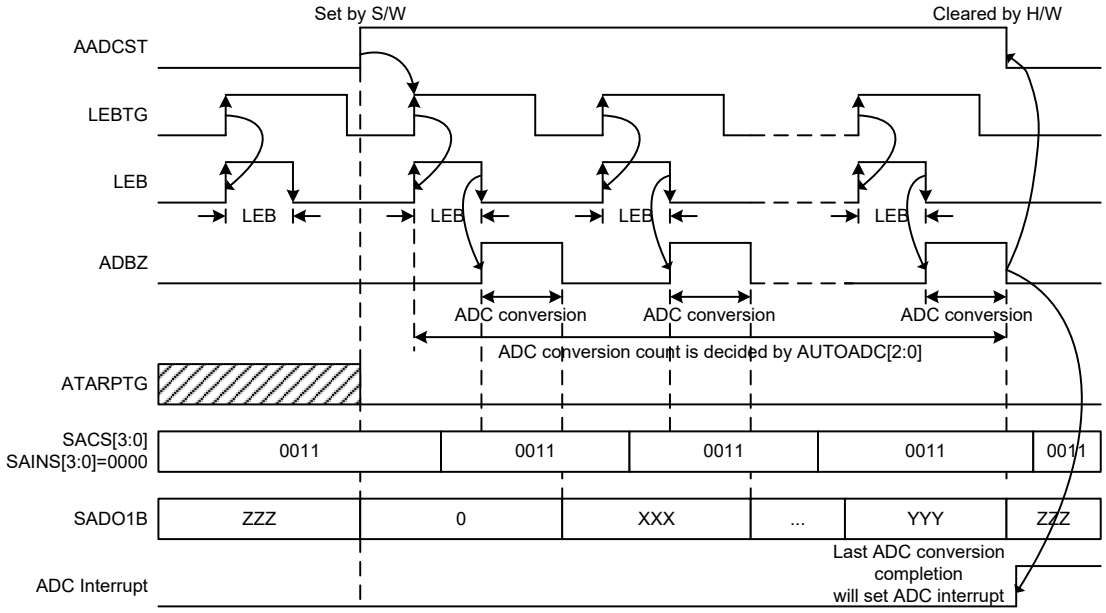
Note that during the automatic conversion process, if the AADCST bit is cleared to 0 when the ADBZ bit is 1, then the automatic conversion circuit will not be reset to its initial state immediately until the current A/D conversion is complete and the ADBZ bit changes to 0. If the AADCST bit

is cleared to 0 when the ADBZ bit is 0 which means the circuit is counting for the LEB time, the counting will stop immediately and the automatic conversion circuit will be reset. After the reset is complete, the AADCST bit can be set high again to enable another group of automatic conversions. If the specified N conversions have been completed when the AADCST bit is cleared, the ADC interrupt can still be triggered.

1-Channel Automatic Conversion

When AADCST changes from 0 to 1, the ATARPTG bit will be cleared to 0 automatically. Then when an LEBTG rising edge trigger signal arrives, the LEB circuit is triggered and the delay timing starts. When the LEB time has elapsed, the A/D converter starts a conversion. After the specified N automatic conversions are completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set high.

In the following example, after an LEBTG rising edge signal arrives, there is enough time for the A/D converter to convert the data before the next LEBTG rising edge arrives.



1-CH Automatic Conversion Timing – 1 (AADCTS[1:0]=00)

Note: 1. AUTOADC[2:0]≠000

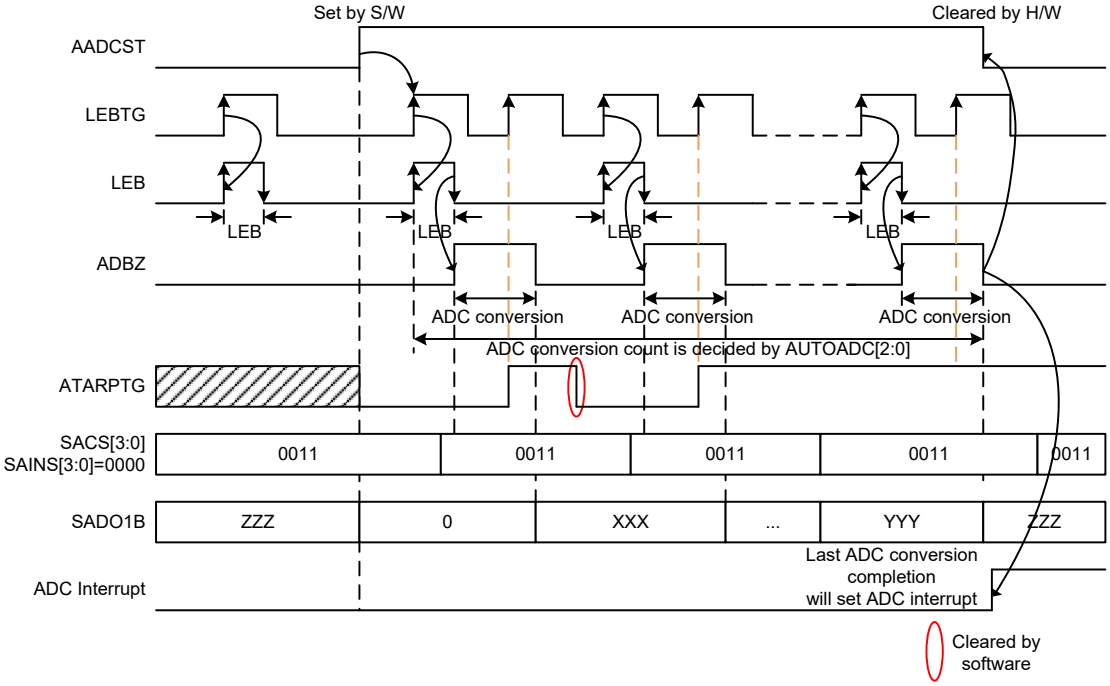
2. XXX is the first A/D converted data

3. YYY is the accumulated A/D conversion result

4. ZZZ is the final accumulated result of the N automatic conversions

When an A/D conversion is in progress (ADBZ=1), if there occurs an LEBTG rising edge, the ATARPTG bit will be set to 1. The A/D converter continues to perform the current conversion, ignoring this trigger signal. After completing the current conversion, the A/D converter will not start the next conversion until a new LEBTG rising edge occurs. After the specified N automatic conversions are completed, the AADCST bit will be cleared by the hardware and the ADC interrupt flag ADF will be set high.

The following shows a 1-channel automatic conversion timing diagram, where an LEBTG rising edge arrives but the previous conversion has not been completed.



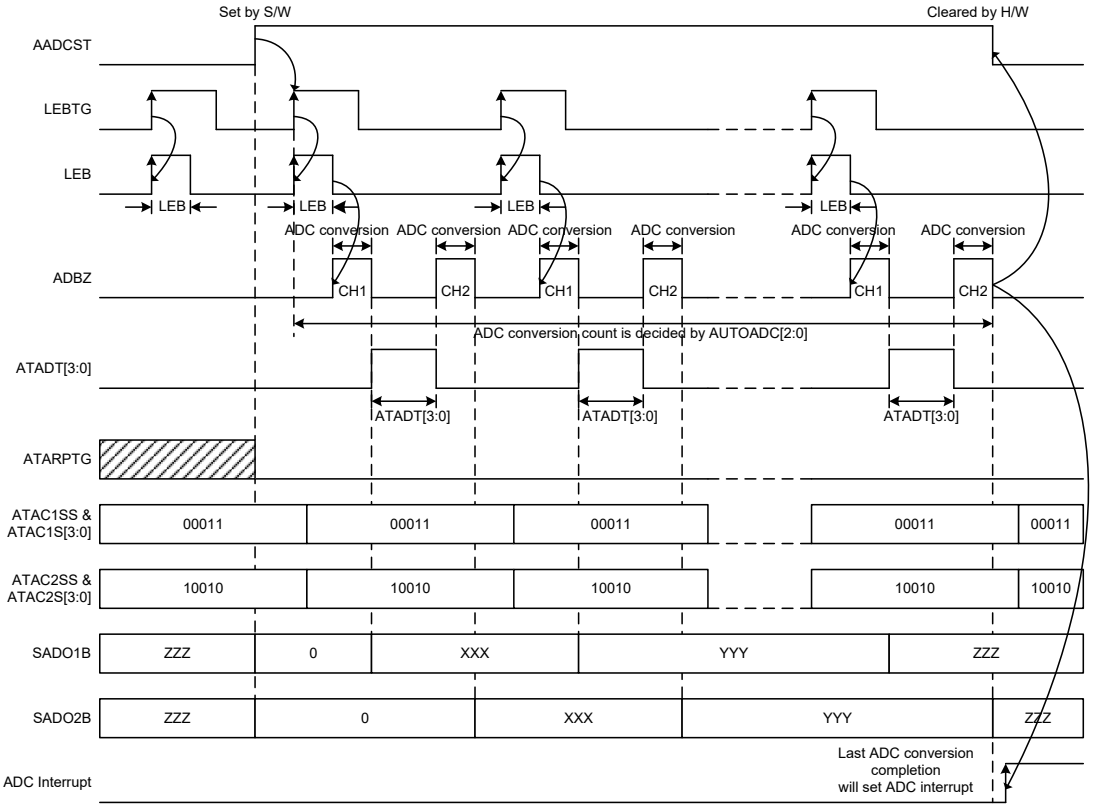
1-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=00)

Note: 1. AUTOADC[2:0]≠000
2. XXX is the first A/D converted data
3. YYY is the accumulated A/D conversion result
4. ZZZ is the final accumulated result of the N automatic conversions

2-Channel Automatic Conversion

When AADCST changes from 0 to 1, the ATARPTG bit will be cleared to 0 automatically. When an LEBTG rising edge trigger signal arrives, the LEB circuit is triggered and the delay timing starts. When the LEB time has elpased, the ADC starts to perform a conversion on the CH1. After completion of the CH1 conversion, a interval time which is defined by the ATADT[3:0] bits is inserted. Then the conversion on CH2 starts. When the CH2 conversion is completed, the A/D converter will wait for the next LEBTG rising edge to start the next cycle of CH1 conversion and CH2 conversion. After N conversions have completed, the AADCST bit will be cleared by the hardware and the ADC interrupt flag ADF will be set high.

In the following example, after an LEBTG rising edge signal arrives, there is enough time for the A/D converter to convert the data of the two channels before the next LEBTG rising edge arrives.



2-CH Automatic Conversion Timing – 1 (AADCTS[1:0]=00)

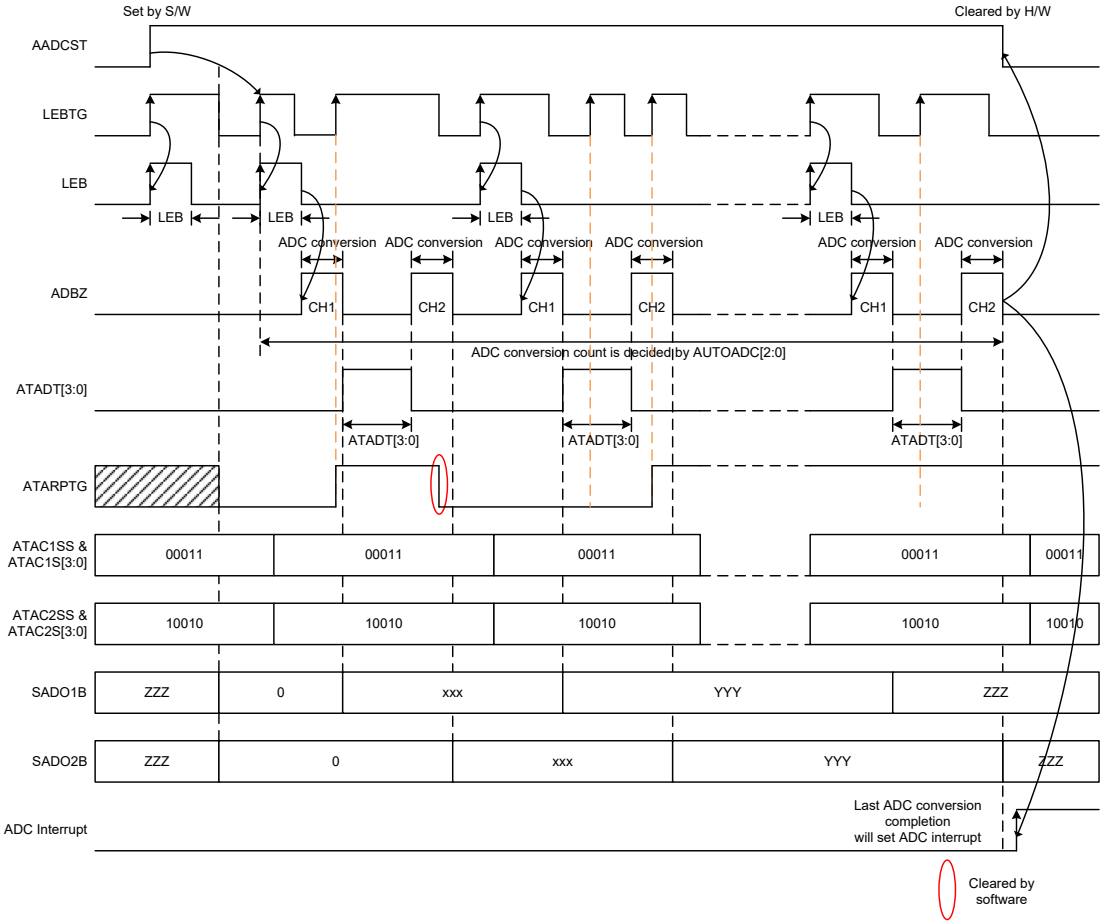
Note: 1. AUTOADC[2:0]≠000

2. XXX is the first A/D converted data

3. YYY is the accumulated A/D conversion result

4. ZZZ is the final accumulated result of the N automatic conversions

In the following example, after an LEBTG rising edge signal arrives, there is no enough time for the ADC to complete the conversions on the two channels before the next LEBTG rising edge arrives. Then the ATARPTG bit will be set high. The A/D converter continues to perform the current conversion, ignoring this trigger signal. After completing the CH1 and CH2 conversions, the A/D converter will not start the next cycle of conversions until a new LEBTG rising edge occurs.



2-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=00)

Note: 1. AUTOADC[2:0]≠000
2. XXX is the first A/D converted data
3. YYY is the accumulated A/D conversion result
4. ZZZ is the final accumulated result of the N automatic conversions

AADCTS[1:0]=01B

When the AADCTS[1:0] bits are 01 and the AUTOADC[2:0] bits are not equal to 000, setting the AADCST bit high will enable the A/D automatic synchronization delay conversion, which is triggered by an OVPINT rising edge activated 16-bit timer counter overflow event. When the TCEXTON bit is 1, after setting the AADCST bit high, a rising edge of the OVPINT signal will set the TCON bit to 1 to enable the timer counter to start counting. The timer will count from the initial value loaded by the preload register, TCR[15:0], to the full count of FFFFH, at which point the timer overflows. The TCON bit is cleared to 0 automatically and an A/D conversion starts. The timer then waits for the next OVPINT signal rising edge. After N automatic conversions have been completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set high. Here the number N is defined by the AUTOADC[2:0] bits. If the trigger source signal is abnormal and no rising edge is generated, the AADCST bit can be cleared using the software.

The AADCST is unable to provide a repeatable trigger. Once the AADCST bit is set high, the circuit cannot be triggered until the N automatic conversions are completed. In program design it should be noted that after the AADCST bit is cleared, the ADF flag should also be cleared using the application progrm.

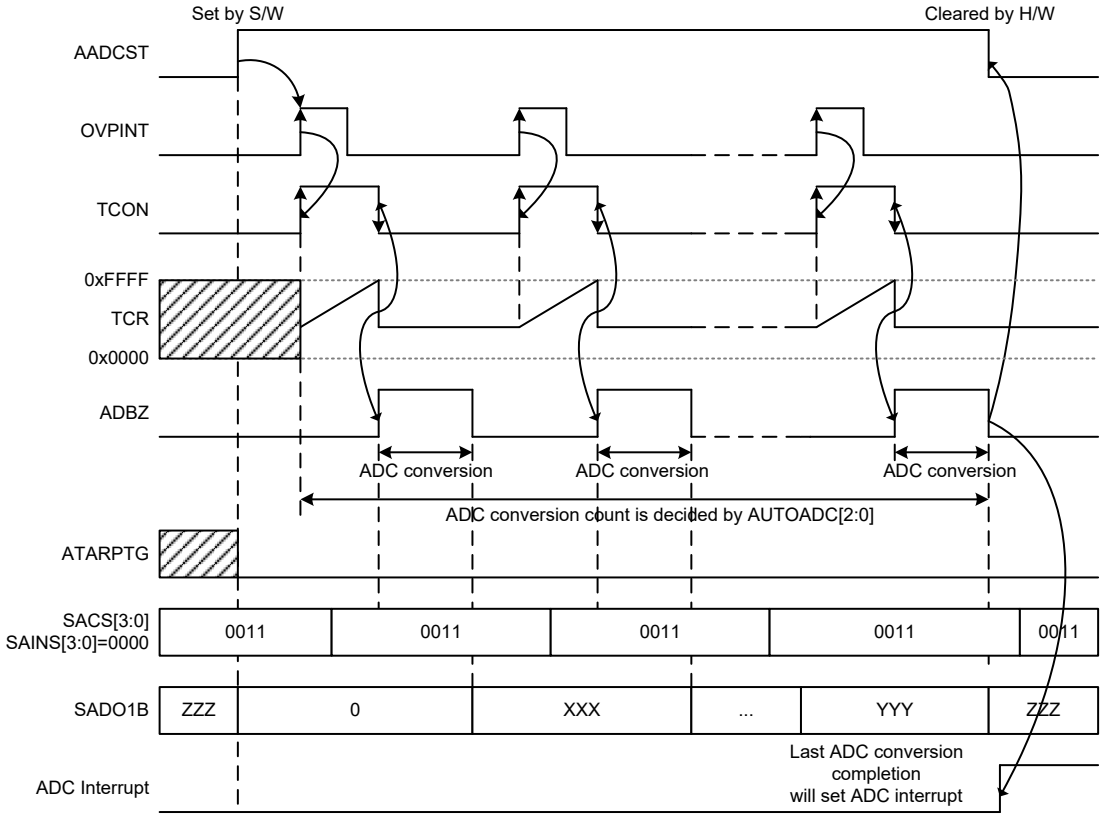
Note that during the automatic conversion process, if the AADCST bit is cleared to 0 when the ADBZ bit is 1, then the automatic conversion circuit will not be reset to its initial state immediately until the current A/D conversion is complete and the ADBZ bit changes to 0. If the AADCST bit is cleared to 0 when the ADBZ bit is 0, the counting will stop immediately and the automatic conversion circuit will be reset. After the reset is completed, the AADCST bit can be set high again to enable another group of automatic conversions. If the specified N conversions have been completed when the AADCST bit is cleared, the ADC interrupt can still be triggered.

Note: if AADCST=0 or if AADCST=1 & TCEXTON=0, an OVPINT signal rising edge will not set the TCON bit to 1.

1-Channel Automatic Conversion

When AADCST changes from 0 to1, the ATARPTG bit will be cleared to 0 automatically. When an OVPINT rising edge signal arrives, the TCON bit will be set high to enable the timer counter to start counting. When the timer counter overflows, the TCON bit is cleared to zero and the A/D converter starts the conversion. After the specified N times of automatic conversions are completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set high.

In the following example, after an OVPINT rising edge signal arrives, there is enough time for the A/D converter to convert the data before the next OVPINT rising edge arrives.



1-CH Automatic Conversion Timing – 1 (AADCTS[1:0]=01)

Note: 1. AUTOADC[2:0]≠000

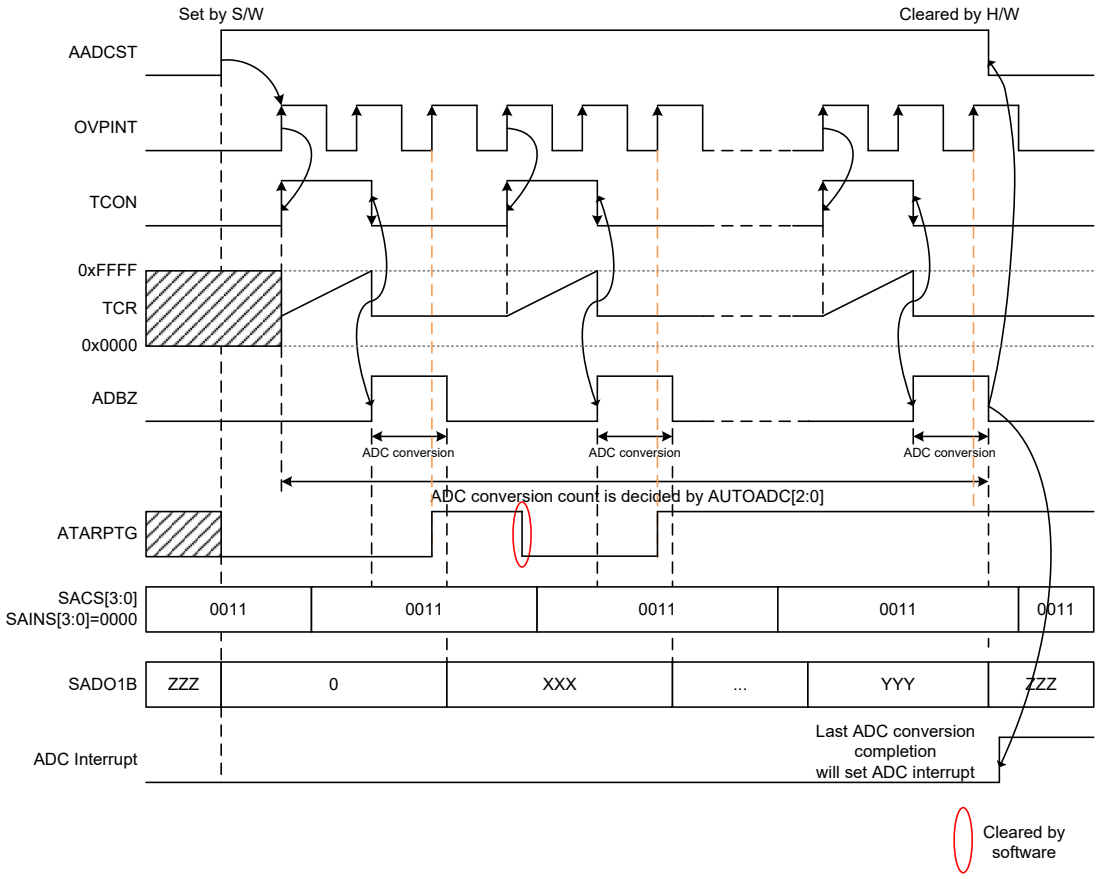
2. XXX is the first A/D converted data

3. YYY is the accumulated A/D conversion result

4. ZZZ is the final accumulated result of the N automatic conversions

In the following example, after AADCST changes from 0 to 1, an OVPINT rising edge will trigger the timer counter to start counting. After the timer overflows, the TCON bit will be cleared to 0 and an A/D conversion starts. There is no enough time for the A/D converter to complete the conversion before the next OVPINT rising edge arrives. Then the ATARPTG bit will be set high. The A/D converter circuitry continues to perform the current conversion, ignoring this OVPINT rising edge trigger signal. After finishing the conversion, the A/D converter will not start the next conversion until a new OVPINT rising edge started timer counter overflows.

Note: When the timer counter is counting-up and has not reached the maximum value, if an OVPINT rising edge occurs, the TCON and ATARPTG bits will not be affected.



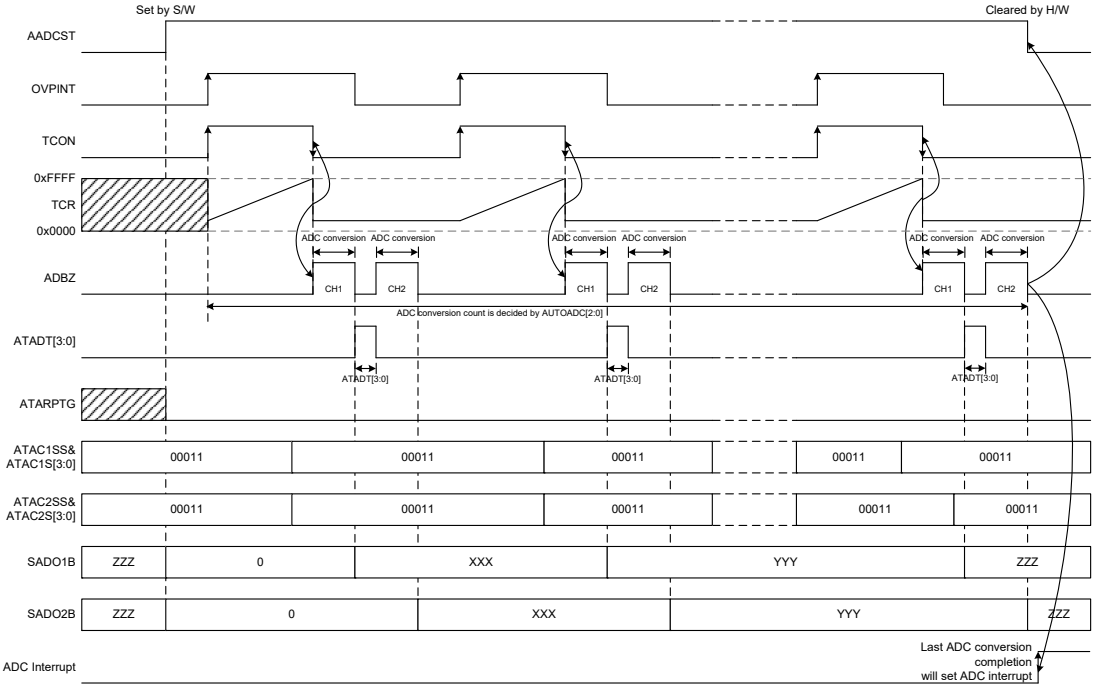
1-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=01)

Note: 1. AUTOADC[2:0]≠000
2. XXX is the first A/D converted data
3. YYY is the accumulated A/D conversion result
4. ZZZ is the final accumulated result of the N automatic conversions

2-Channel Automatic Conversion

After changing the AADCST bit from low to high, the ATARPTG bit will be cleared to 0 automatically. A rising edge of the OVPINT signal will set the TCON bit to 1 to enable the timer counter to start counting up. Once the timer counter overflows, the TCON bit will be cleared and an A/D conversion on CH1 will start. After completion of the CH1 conversion, a interval time which is defined by the ATADT[3:0] bits is inserted. Then the A/D conversion on CH2 starts. After the CH2 conversion has completed, an additional OVPINT rising edge will be required. When the number of conversions on CH1 and CH2 reaches the set value N, the AADCST bit will be automatically cleared by the hardware and the ADC interrupt flag ADF will be set high.

The following shows that the A/D converter has enough time to complete the conversion on CH1 and CH2 before the next OVPINT rising edge arrives.



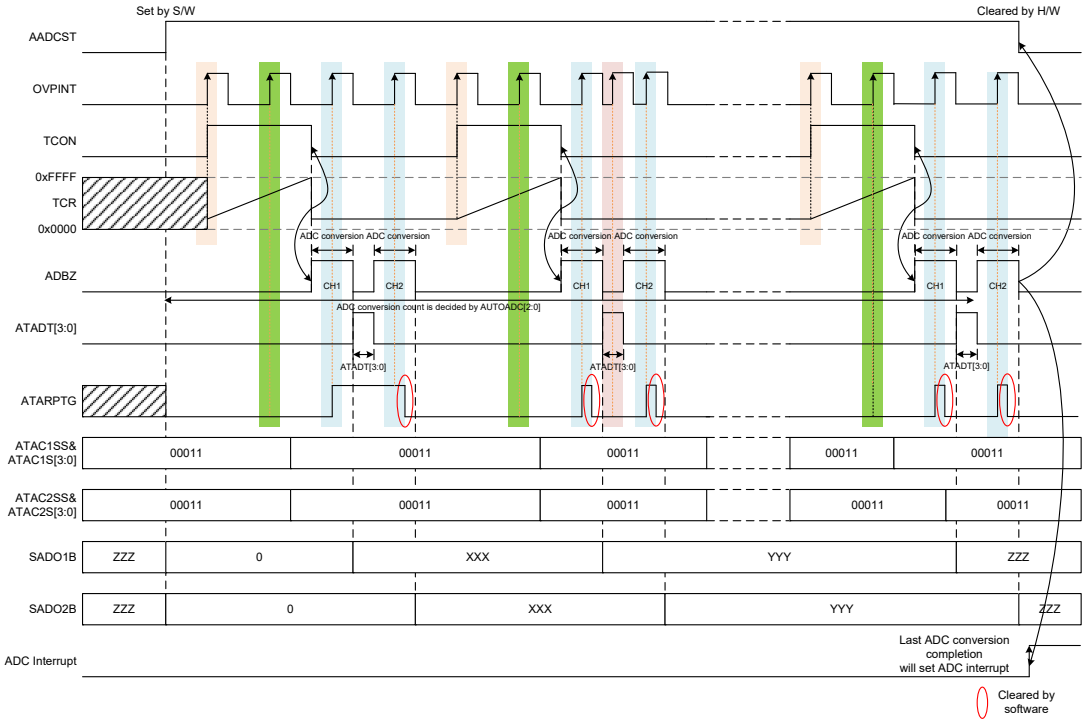
2-CH Automatic Conversion Timing – 1 (AADCTS[1:0]=01)

Note: 1. AUTOADC[2:0]≠000

2. XXX is the first A/D converted data
3. YYY is the accumulated A/D conversion result
4. ZZZ is the final accumulated result of the N automatic conversions

In the following timing diagram, after AADCST changes from 0 to 1, an OVPINT rising edge will trigger the timer counter to start counting. After the timer overflows, the TCON bit is cleared to 0 and an A/D conversion is started. However there is no enough time for the A/D converter to complete the conversions on CH1 and CH2 before the next OVPINT rising edge arrives, then the ATARPTG bit will be set high. The A/D converter circuitry continues to perform the current conversion, ignoring this OVPINT rising edge trigger signal. After completing the CH1 and CH2 conversions, the A/D converter will not start the next conversion until a new coming OVPINT rising edge started timer counter overflows, and then converts the next data until all configured conversions are completed.

Note: When the timer counter is counting-up and has not reached the maximum value, if an OVPINT rising edge occurs, the TCON and ATARPTG bits will not be affected.



2-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=01)

Note: 1. AUTOADC[2:0]≠000

2. XXX is the first A/D converted data

3. YYY is the accumulated A/D conversion result

4. ZZZ is the final accumulated result of the N automatic conversions

AADCTS[1:0]=10B

When the AADCTS[1:0] bits are 10 and the AUTOADC[2:0] bits are not equal to 000, setting the AADCST bit high will enable the A/D automatic synchronization delay conversion, which is triggered by an AADCST rising edge activated 16-bit timer counter overflow event. When setting the AADCST bit high, the TCON bit will also be set to 1 by the hardware to enable the timer counter to start counting. The timer counter will count from the initial value loaded by the preload register, TCR[15:0], to the full count of FFFFH, at which point the timer counter overflows, and an A/D conversion is started. The timer counter counts from the reload value again and when it overflows, the next conversion will be triggered. After N conversions have been completed, the TCON and AADCST bits will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set high. Here the number N is defined by the AUTOADC[2:0] bits. If the timer counter is abnormal or the TCON bit is cleared, the AADCST bit can be cleared using the software.

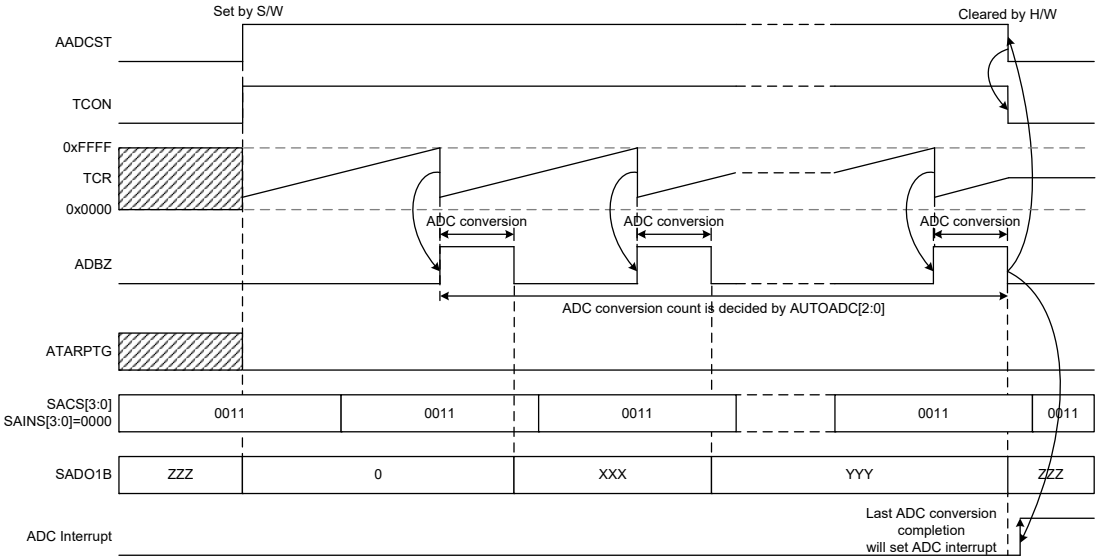
The AADCST bit is unable to provide a repeatable trigger. Once the AADCST bit is set high, the circuit cannot be triggered until the specified N automatic conversions are completed. In program design it should be noted that after the AADCST bit is cleared, the ADF flag should also be cleared using the application progrm.

Note that during the automatic conversion process, if the AADCST bit is cleared to 0 when the ADBZ bit is 1, then the automatic conversion circuit will not be reset to its initial state immediately until the current A/D conversion is complete and the ADBZ bit changes to 0. If the AADCST bit is cleared to 0 when the ADBZ bit is 0, the counting will stop immediately and the automatic conversion circuit will be reset. After the reset is completed, the AADCST bit can be set high again to enable another group of automatic conversions. If the specified N conversions have been completed when the AADCST bit is cleared, the ADC interrupt can still be triggered.

1-Channel Automatic Conversion

When AADCST changes from 0 to1, the ATARPTG bit will be cleared to 0 automatically and the TCON bit is set to 1 to enable the timer counter to start counting up. When the timer counter overflows, the A/D converter starts a conversion and the timer counter restarts counting from the preload value. After the specified N times of automatic conversions are completed, the AADCST and TCON bits will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set high.

In the following example, after an A/D conversion is started by the timer counter overflow event, there is enough time for the ADC to complete a data conversion before the timer counter overflows again.

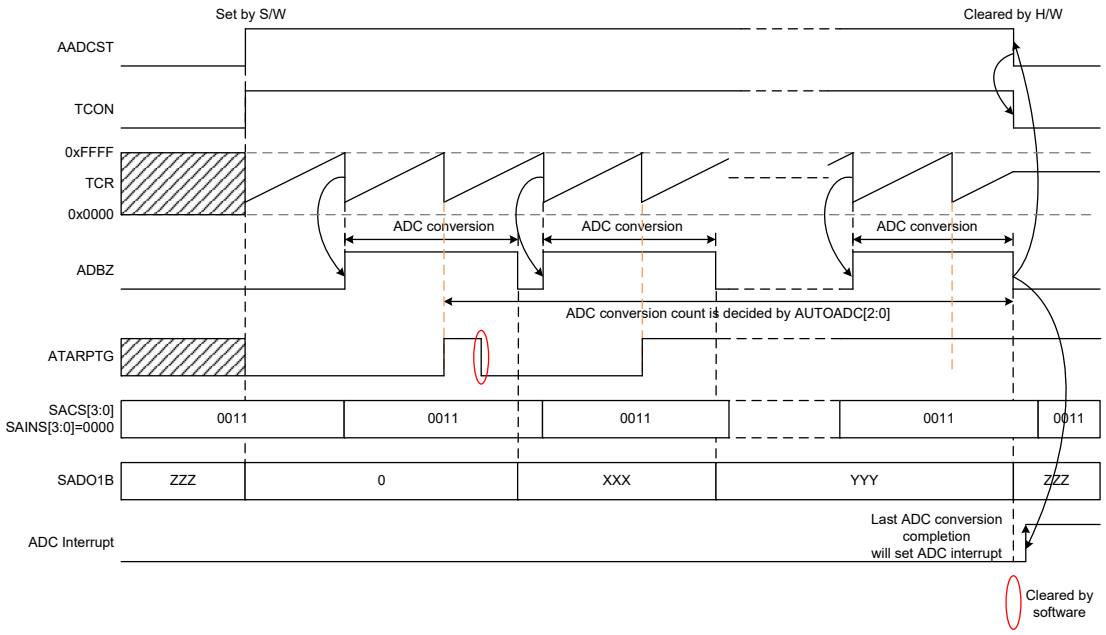


1-CH Automatic Conversion Timing – 1 (AADCTS[1:0]=10)

Note: 1. AUTOADC[2:0]≠000

2. XXX is the first A/D converted data
3. YYY is the accumulated A/D conversion result
4. ZZZ is the final accumulated result of the N automatic conversions
5. The TCR counter must be written with a new value when TCON=0, otherwise it will continue counting from the previous final counter value when TCON=1.

In the following timing diagram, after an A/D conversion is started by the timer counter overflow event, there is no enough time for the ADC to complete the conversion before the timer counter overflows again. Then the ATARPTG bit will be set high. The A/D converter circuitry continues to perform the current conversion, ignoring this overflow event. After completing the conversion, the A/D converter will not start the next conversion until a new timer counter overflow occurs, and then convert the next data until all configured conversions are completed.



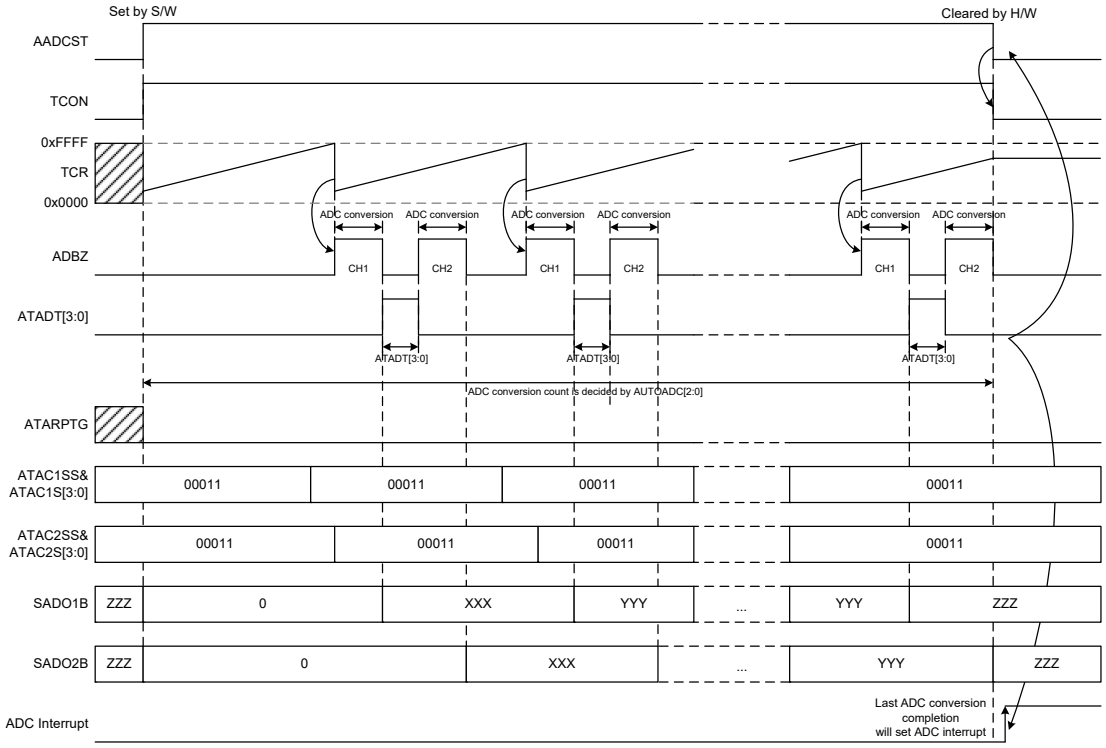
1-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=10)

Note: 1. AUTOADC[2:0]≠000
2. XXX is the first A/D converted data
3. YYY is the accumulated A/D conversion result
4. ZZZ is the final accumulated result of the N automatic conversions
5. The TCR counter must be written with a new value when TCON=0, otherwise it will continue counting from the previous final counter value when TCON=1.

2-Channel Automatic Conversion

When setting the AADCST bit high, the ATARPTG bit will be cleared to 0 automatically and the TCON bit will also be set to 1 by the hardware to enable the timer counter to start counting. The timer counter will count from the initial value loaded by the preload register, TCR[15:0], to the full count of FFFFH, at which point the timer overflows and restarts counting up from the reload value. At the same time an A/D conversion on CH1 is started. After completion of the CH1 conversion, an interval time which is defined by the ATADT[3:0] bits is required. Then an A/D conversion on CH2 starts. After this CH2 conversion has completed, the circuit will wait for the next timer counter overflow event to start the next cycle of conversions. When the number of conversion cycles on CH1 and CH2 reaches the set value N, the hardware will clear the AADCST and TCON bits automatically and the ADC interrupt flag ADF will be set high.

In the following timing diagram, after an A/D conversion is started by the timer counter overflow event, the A/D converter has enough time to complete the conversion on CH1 and CH2 before the timer counter overflows again, and then converts the next data until all configured conversions are completed.

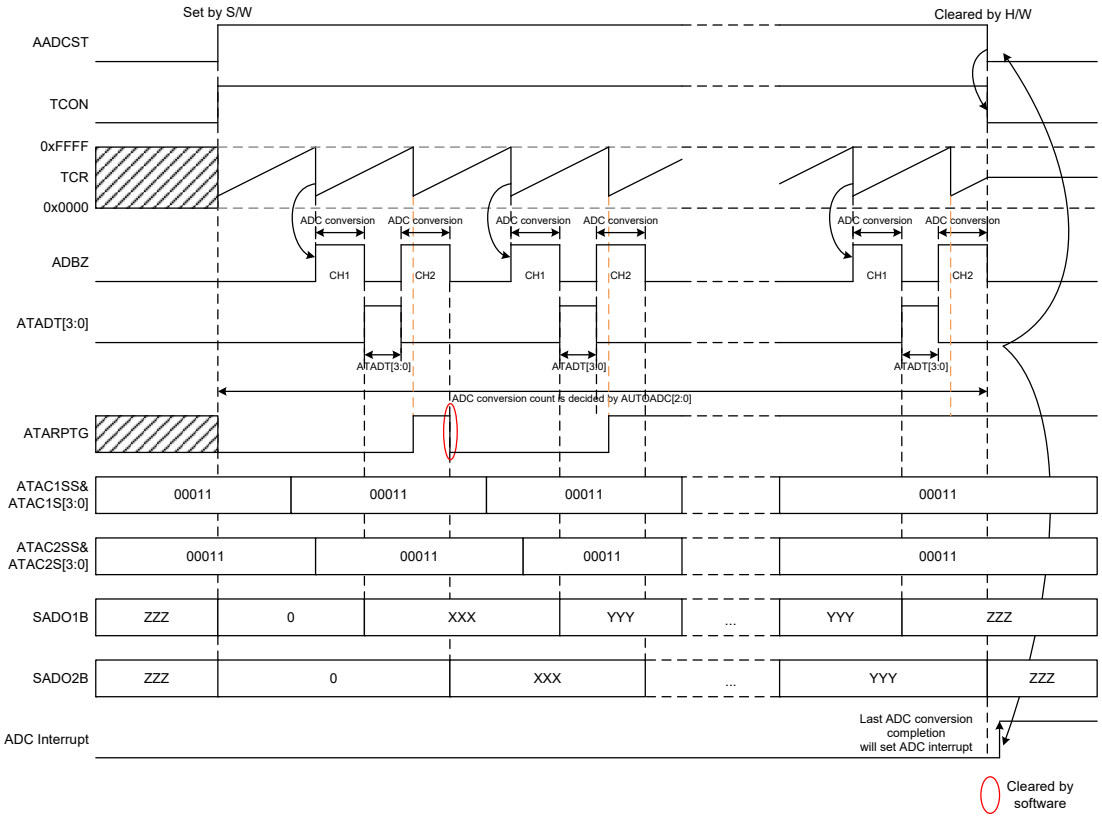


2-CH Automatic Conversion Timing – 1 (AADCTS[1:0]=10)

Note: 1. AUTOADC[2:0]≠000

2. XXX is the first A/D converted data
3. YYY is the accumulated A/D conversion result
4. ZZZ is the final accumulated result of the N automatic conversions
5. The TCR counter must be written with a new value when TCON=0, otherwise it will continue counting from the previous final counter value when TCON=1.

In the following timing diagram, after an A/D conversion is started by the timer counter overflow event, there is no enough time for the A/D converter to complete the conversions on CH1 and CH2 before the timer counter overflows again, then the ATARPTG bit will be set high. The A/D converter circuitry continues to perform the current conversion, ignoring this counter overflow event. After finishing the CH1 and CH2 conversions, the A/D converter will not start the next conversion until a new timer counter overflow event occurs, and then convert the next data until all configured conversions are complete.



2-CH Automatic Conversion Timing – 2 (AADCTS[1:0]=10)

Note: 1. AUTOADC[2:0]≠000
2. XXX is the first A/D converted data
3. YYY is the accumulated A/D conversion result
4. ZZZ is the final accumulated result of the N automatic conversions
5. The TCR counter must be written with a new value when TCON=0, otherwise it will continue counting from the previous final counter value when TCON=1.

AADCTS[1:0]=11B

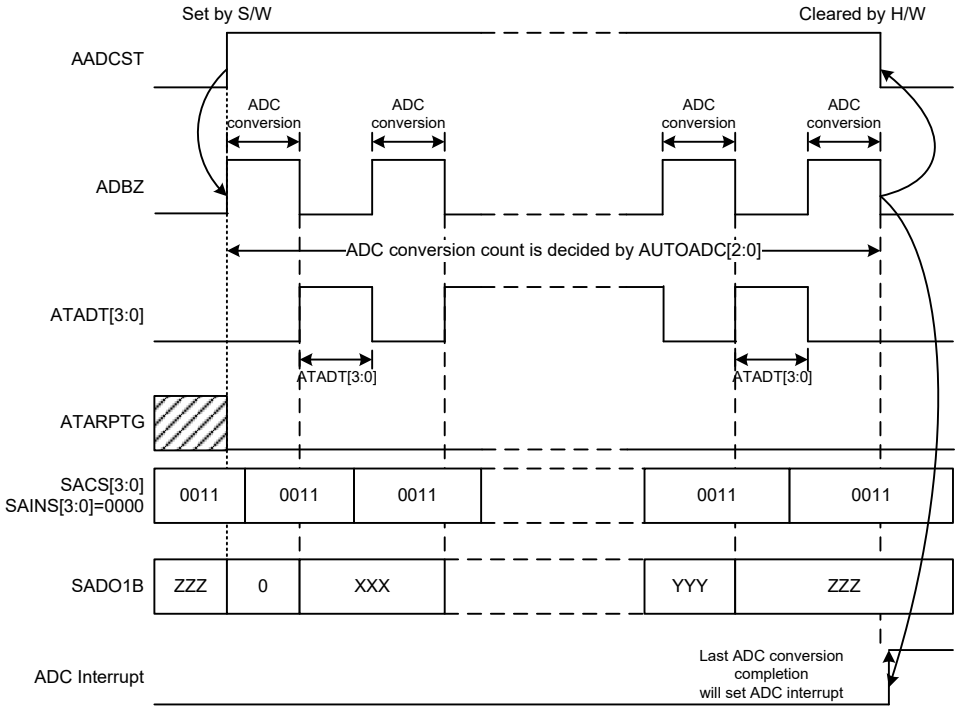
When the AADCTS[1:0] bits are 11 and the AUTOADC[2:0] bits are not equal to 000, setting the AADCST bit high will start an A/D conversion immediately. When one conversion is complete, after a period of time which is defined by the ATADT[3:0] bits, the next conversion starts. After N conversions have been completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set high. Here the number N is defined by the AUTOADC[2:0] bits.

The AADCST is unable to provide a repeatable trigger. Once the AADCST bit is set high, the circuit cannot be triggered until the specified N automatic conversions are completed. In program design it should be noted that after the AADCST bit is cleared, the ADF flag should also be cleared using the application progrm.

Note that during the automatic conversion process, if the AADCST bit is cleared to 0 when the ADBZ bit is 1, then the automatic conversion circuit will not be reset to its initial state immediately until the current A/D conversion is complete and the ADBZ bit changes to 0. If the AADCST bit is cleared to 0 when the ADBZ bit is 0, the automatic conversion circuit will be reset immediately. After the reset is complete, the AADCST bit can be set high again to enable another group of automatic conversions. If the specified N conversions have been completed when the AADCST bit is cleared, the ADC interrupt can still be triggered.

1-Channel Automatic Conversion

When AADCST changes from 0 to 1, the ATARPTG bit will be cleared to 0 automatically and an A/D automatic conversion starts. After the specified N automatic conversions are completed, the AADCST bit will be cleared to zero by the hardware and the ADC interrupt flag ADF will be set high.

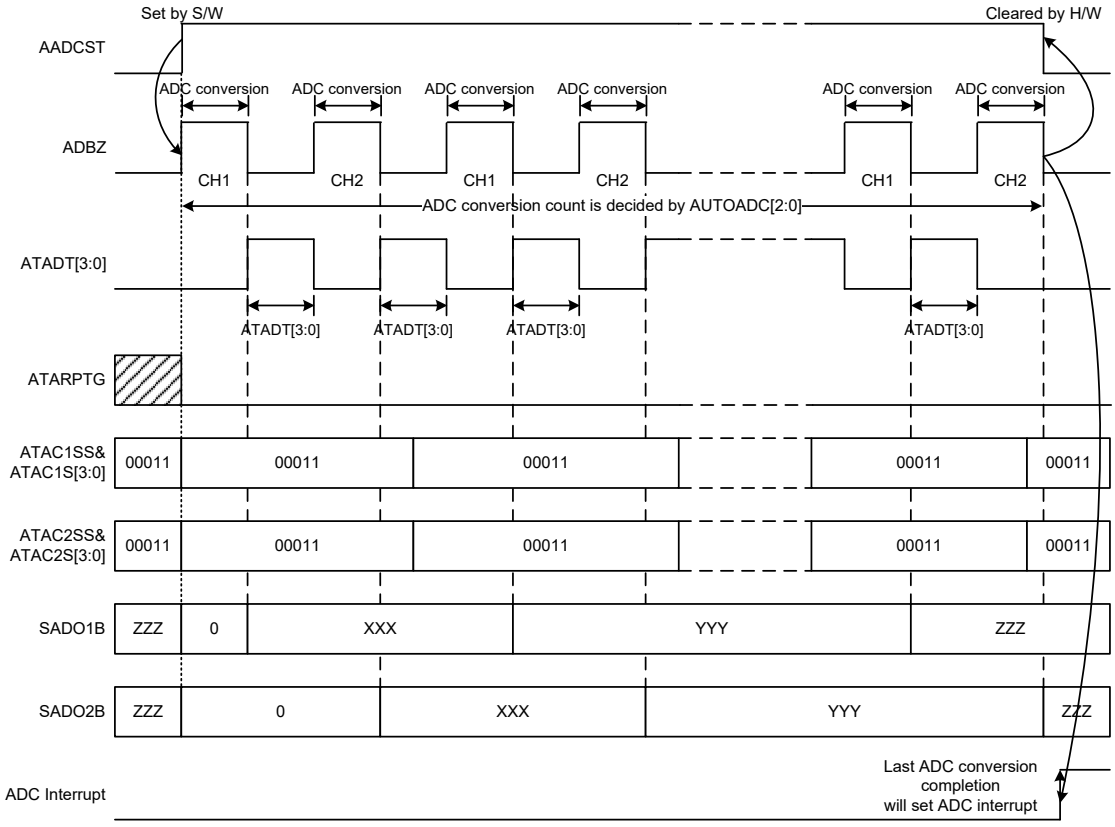


1-CH Automatic Conversion Timing (AADCTS[1:0]=11)

Note: 1. AUTOADC[2:0]≠000
2. XXX is the first A/D converted data
3. YYY is the accumulated A/D conversion result
4. ZZZ is the final accumulated result of the N automatic conversions

2-Channel Automatic Conversion

When setting the AADCST bit from 0 to 1, the ATARPTG bit will be cleared to 0 automatically and an A/D conversion on CH1 is started. After completion of the CH1 conversion, an interval time which is defined by the ATADT[3:0] bits is required. Then an A/D conversion on CH2 starts. After this CH2 conversion is completed and an interval time defined by the ATADT[3:0] bits has passed, the next cycle of conversions on CH1 and CH2 starts again. When the number of conversion cycles on CH1 and CH2 reaches the set value N, the hardware will clear the AADCST bit automatically and the ADC interrupt flag ADF will be set.



2-CH Automatic Conversion Timing (AADCTS[1:0]=11)

Note: 1. AUTOADC[2:0]≠000

2. XXX is the first A/D converted data

3. YYY is the accumulated A/D conversion result

4. ZZZ is the final accumulated result of the N automatic conversions

Programming Considerations

When the ADCEN bit in the SADC0 register is set high to power on the A/D conversion internal circuitry, a certain delay must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D converter inputs by configuring the corresponding pin control bits, if the ADCEN bit is high then some power will still be consumed.

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

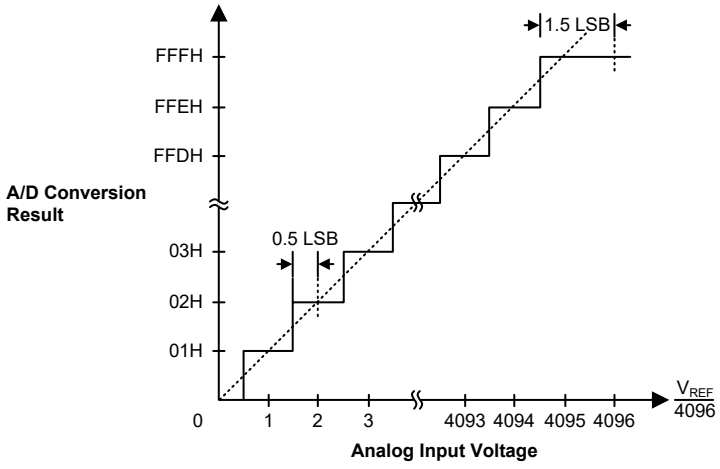
$$1\ LSB=V_{REF}\div 4096$$

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

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

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 SAVRS bit field.



Ideal A/D Conversion Function

Temperature Sensor Function

The A/D converter integrates a temperature sensor circuit. After the temperature sensor output signal is input to the A/D converter for conversion, the current ambient temperature value can be obtained using a specific formula. When measuring the temperature sensor output voltage, it is recommended to use the V_{VR} reference voltage of 1.6V.

- Step 1

  Select V_{TSO} as the A/D converter input (SAINS[3:0]=1011B) and V_{VR} as the A/D converter reference voltage (SAVRS[1:0]=10B/11B, PGAGS[1:0]=01B, PGAIS[1:0]=01B).

- Step 2

  Enable the temperature sensor (TSEN=1). A time of t_{TSS} should be allowed for temperature sensor to stablise.

- Step 3

  Since the large output impedance of the temperature sensor, the A/D conversion sampling time t_{ADS} should be greater than or equal to16μs, it is recommended to properly configure the t_{ADCK} in a range of 4μs~10μs (typical 8μs).

- Step 4

  Start A/D conversion and read the converted value. It is recommended to read eight times and take the average value, then record it as ADC_{TS}.

- Step 5

  The measured temperature value is obtained using the following formula:

$$Ta\ (°C) = 0.0963 \times ADC_{TS} - 281.06 + T_{OS}$$

  Where the T_{OS} value has been previousely calibrated by writer and stored in the Option Memory.

Note: 1. Users should follow the above steps to measure Ta under a known temperature condition. The measured Ta value minus the known temperature value is a new T_{OS} value. Then store it in the Option Memory to correct the offset of the formula.

2. The T_{OS} format stored in the Option Memory is 2's complement, 7 bits for the integer including the sign bit and 1 bit for the decimal.

3. When using the writer for programming, a dedicated temperature module (EMDE001A) developed by Holtek must be externally connected to obtain the measurement data of TS (ambient temperature), ADC_{TS} (converted data of TS) and T_{OS} (calibrated compensation value), which are then stored in the Option Memory by writer. They can be read from the Program Memory last page using the table read instruction when the Option Memory mapping function is enabled.

Name	Mapped Address in Program Memory	Description
TS	3FD3H	TS code (00H(0°C)~FFH(51°C)) Temperature value can be converted from the code with 0.2°C/step
ADC_{TS}	3FD4H	12-bit TS A/D converted value bit 11 ~ bit 0
T_{OS}	3FD5H	Temperature sensor single point calibration compensation value

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 Converter Programming Examples

The following two programming examples illustrate how to setup and implement a manual trigger 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 converter interrupt is used to determine when the conversion is complete.

Example 1: using an ADBZ polling method to detect the end of conversion

```
mov a,00h               ; disable A/D automatic conversion function
mov SADC4,a
clr ADE                 ; disable ADC interrupt
mov a,03h               ; select fSYS/8 as A/D clock and A/D input signal
                        ; comes from external channel
mov SADC1,a
mov a,00h,              ; select AVDD as A/D reference voltage source
mov SADC2,a
mov a,03h               ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,20h               ; enable A/D converter and select AN0 external channel input
mov SADC0,a
:
start_conversion:
clr START               ; high pulse on start bit to initiate conversion
set START               ; reset A/D converter
clr START               ; start A/D conversion
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 2: using the interrupt method to detect the end of conversion

```
mov a,00h               ; disable A/D automatic conversion function
mov SADC4,a
clr ADE                 ; disable ADC interrupt
mov a,03h               ; select fSYS/8 as A/D clock and A/D input signal
                        ; comes from external channel
mov SADC1,a
mov a,00h,              ; select AVDD as A/D reference voltage source
mov SADC2,a
mov a,03h               ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,20h               ; enable A/D converter and select AN0 external channel input
mov SADC0,a
:
Start_conversion:
clr START               ; high pulse on START bit to initiate conversion
set START               ; reset A/D converter
clr START               ; start A/D conversion
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
```

Serial Interface Module – SIM

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

SPI Interface

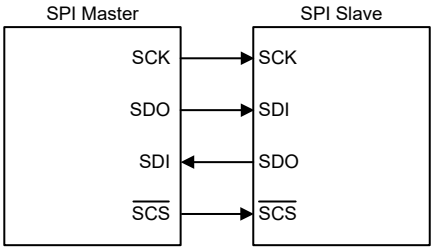
The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices, etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the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

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

SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four-line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, SCK is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I^2C function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. After the desired SPI configuration has been set it can be disabled or

enabled using the SIMEN bit in the SIMC0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ pin only one slave device can be utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to 1 to enable $\overline{SCS}$ pin function, set CSEN bit to 0 the $\overline{SCS}$ pin will be floating state.

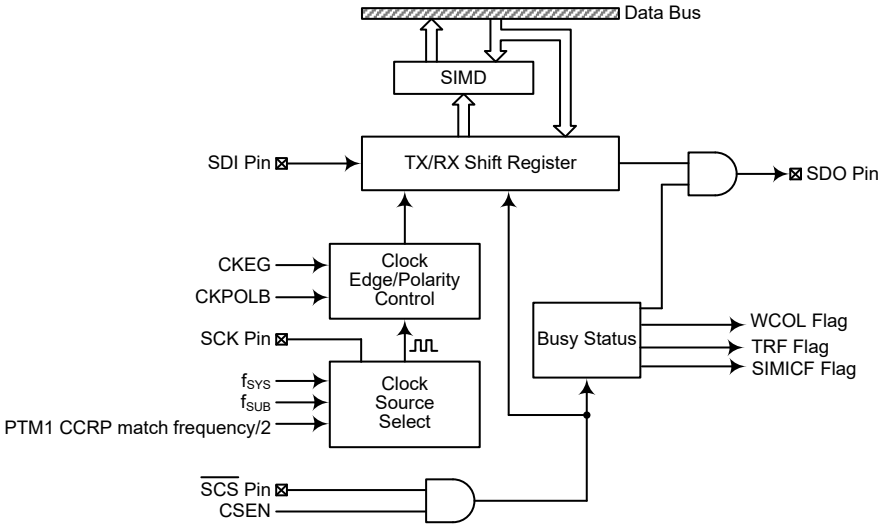


SPI Master/Slave Connection

The SPI function in this device offers the following features:

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

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



SPI 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 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 SIMC1 register is not used by the SPI function, only by the I²C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Register SIMC2 is used for other control functions such as LSB/MSB selection, write collision flag, etc.

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is PTM1 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Non SIM function

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 PTM1 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

The SIMDEB1~SIMDEB0 bits are only used in the I²C mode and the detailed definition is described in the I²C 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 slave mode Incomplete Transfer Flag

0: SIM SPI slave mode incomplete condition not occurred
1: SIM SPI slave mode incomplete condition occurred

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

- **SIMC2 Register**

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

Bit 7~6 Undefined bits

These bits can be read or written by the application program.

Bit 5 **CKPOLB**: SPI clock line base condition selection

0: The SCK line will be high when the clock is inactive
1: The SCK line will be low when the clock is inactive

The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4 **CKEG**: SPI SCK clock active edge type selection

CKPOLB=0

0: SCK is high base level and data capture at SCK rising edge
1: SCK is high base level and data capture at SCK falling edge

CKPOLB=1

0: SCK is low base level and data capture at SCK falling edge
1: SCK is low base level and data capture at SCK rising edge

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

will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOLB bit.

Bit 3 **MLS**: SPI data shift order
0: LSB first
1: MSB first
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2 **CSEN**: SPI $\overline{SCS}$ pin control
0: Disable
1: Enable
The CSEN bit is used as an enable/disable for the $\overline{SCS}$ pin. If this bit is low, then the $\overline{SCS}$ pin will be disabled and placed into a floating condition. If the bit is high, the $\overline{SCS}$ pin will be enabled and used as a select pin.

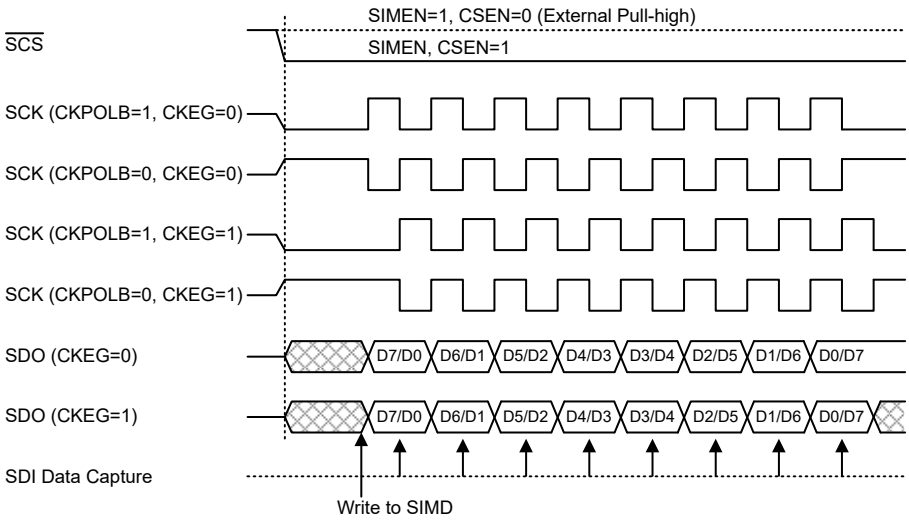
Bit 1 **WCOL**: SPI write collision flag
0: No collision
1: Collision
The WCOL flag is used to detect whether a data collision has occurred or not. If this bit is high, it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will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

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

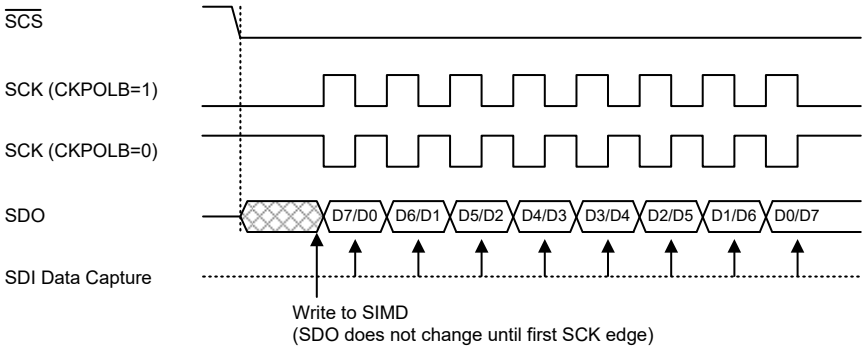
SPI Communication

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 diagrams show the relationship between the slave data and SCK signal for various configurations of the CKPOLB and CKEG bits.

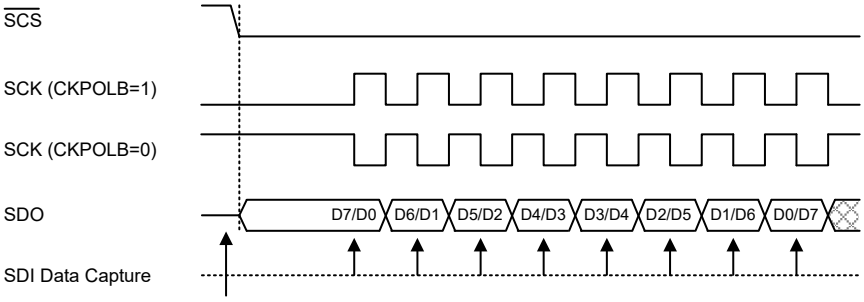
The SPI Master mode will continue to function if the SPI clock is running.



SPI Master Mode Timing

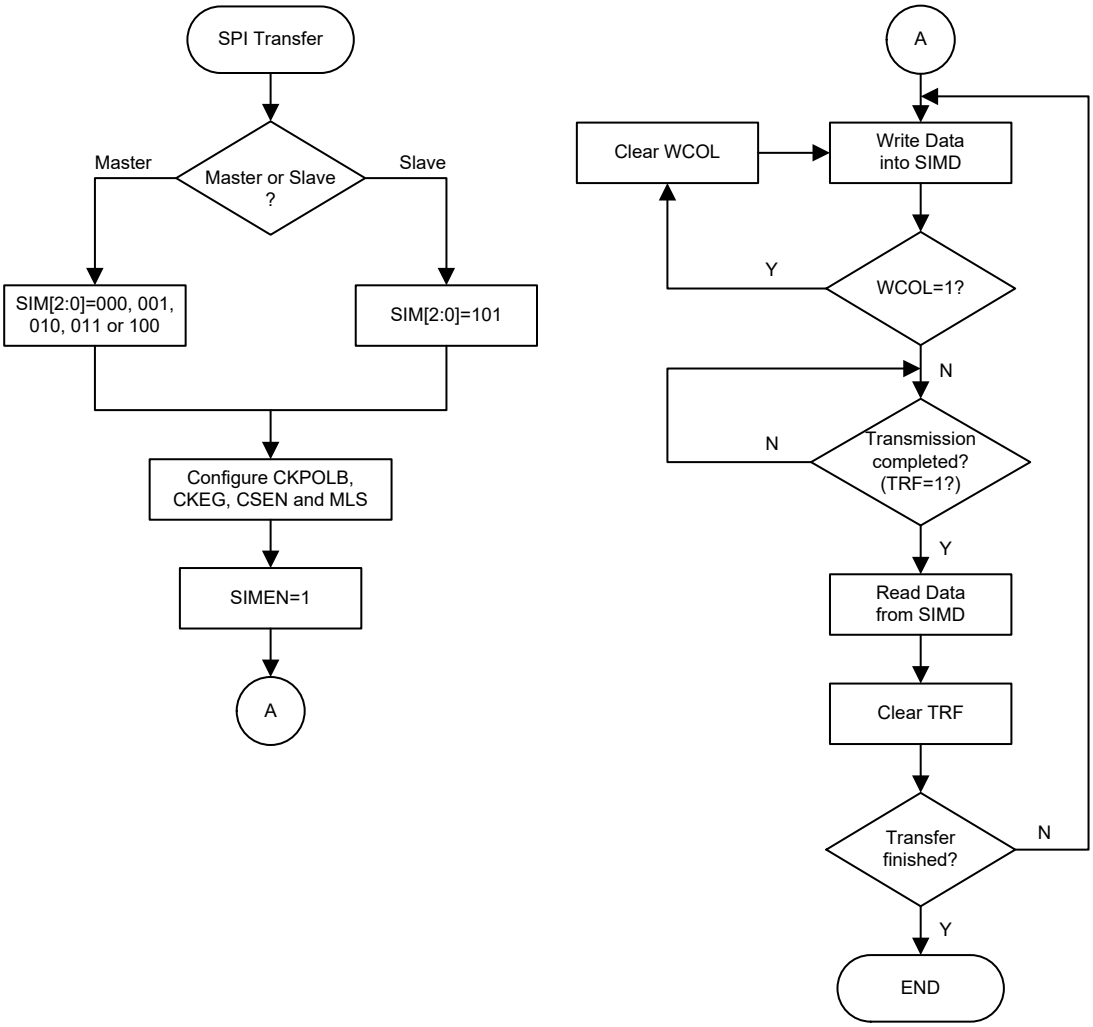


SPI Slave Mode Timing – CKEG=0



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

SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flow Chart

SPI Bus Enable/Disable

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

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

SPI Operation Steps

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

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

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

Master Mode

- Step 1

  Select the SPI Master mode and clock source using the SIM2~SIM0 bits in the SIMC0 control register.

- Step 2

  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave devices.

- Step 3

  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.

- Step 4

  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then use the SCK and SDO lines to output the data. After this, go to step 5.

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

- Step 5

  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.

- Step 6

  Check the TRF bit or wait for 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 step 5.

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

- Step 5

  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.

- Step 6

  Check the TRF bit or wait for a SIM SPI serial bus interrupt.

- Step 7

  Read data from the SIMD register.

- Step 8

  Clear TRF.

- Step 9

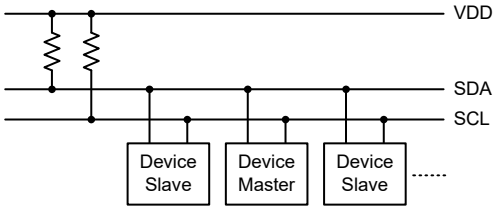
  Go to step 4.

Error Detection

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

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

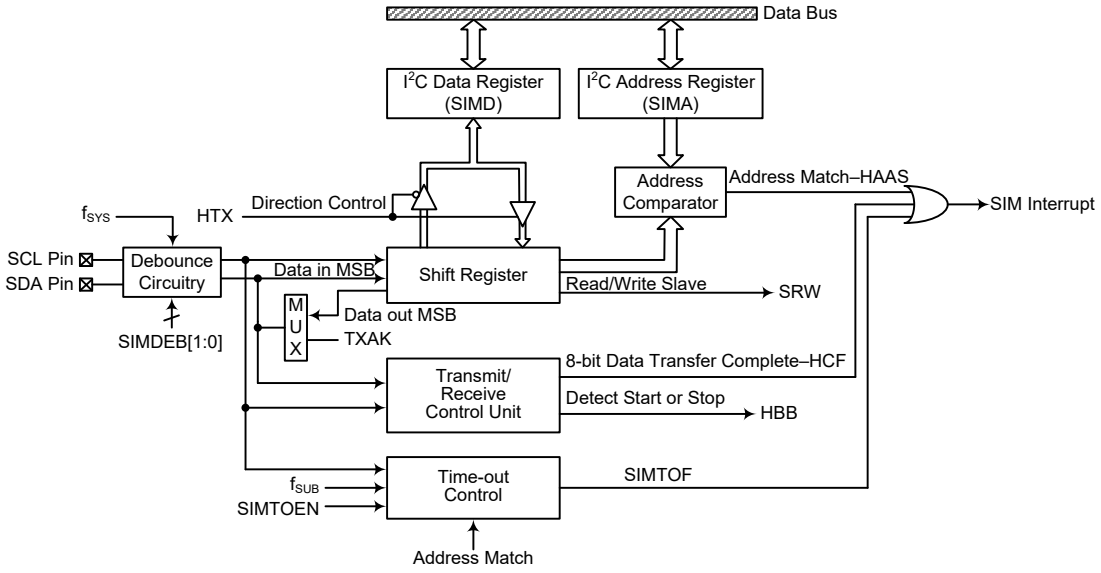
The I²C serial interface is a two-line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. 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

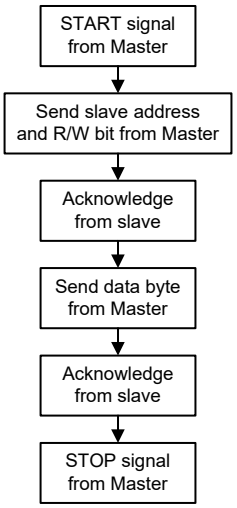
operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode.

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.

It is suggested that the device should not enter the IDLE/SLEEP mode during the I²C communication.



I²C Interface Block Diagram



I²C Interface Operation

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

I²C Debounce Time Selection	I²C Standard Mode (100kHz)	I²C Fast Mode (400kHz)
No Debounce	f_{SYS}>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 Requirement

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
SIMA	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

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 functions. 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 of the slave device is stored. Bits 7~1 of the SIMA register define the device slave address. Bit 0 is not implemented.

When a master device, which is connected to the 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 locates at 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 bit, can be read or written

I²C Control Registers

There are three control registers for the I²C interface, SIMC0, SIMC1 and SIMTOC. The register SIMC0 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 PTM1 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Non SIM function

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 PTM1 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. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I²C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

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

The SIMICF bit is only used in the SPI mode and the detailed definition is described in the SPI section.

• **SIMC1 Register**

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

Bit 7 **HCF**: 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 completion flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Bit 6 **HAAS**: I²C Bus data transfer completion flag
0: Not address match
1: Address match

The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB**: I²C Bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy

The HBB flag is the I²C busy flag. This flag will be "1" when the I²C bus is busy which will occur when a START signal is detected. The flag will be cleared to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX**: I²C slave device transmitter/receiver selection
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK**: I²C bus transmit acknowledge flag
0: Slave sends acknowledge flag
1: Slave does not send acknowledge flag

The TXAK flag is the transmit acknowledge flag. After the slave device has received 8 bits of data, this flag will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set the 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 matches the slave address, 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 control
0: Disable
1: Enable – must be cleared by the application program after wake-up

This bit should be set to 1 to enable the I²C address match wake-up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake-up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

Bit 0 **RXAK**: I²C bus receive acknowledge flag
0: Slave receives acknowledge flag
1: Slave does not receive acknowledge flag

The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that an acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device is in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the 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 interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from either an address match or the completion of an 8-bit data transfer or the 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

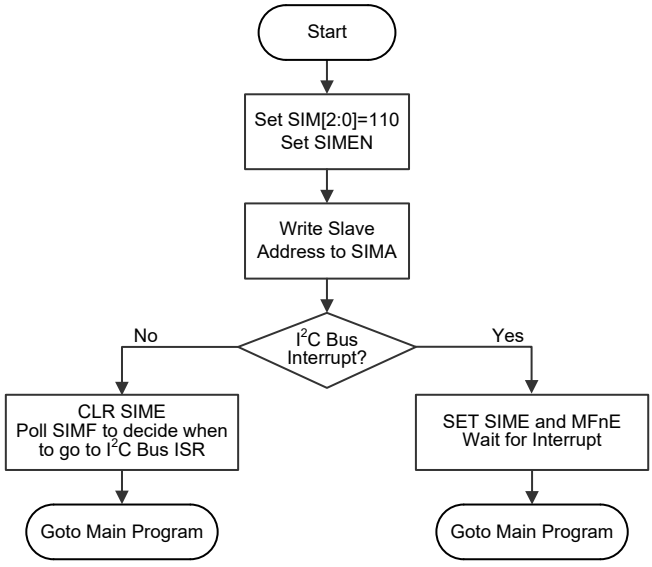
  Set the SIM2~SIM0 bits to "110" and SIMEN bit to "1" in the SIMC0 register to enable the I²C bus.

- Step 2

  Write the slave address of the device to the I²C bus address register SIMA.

- Step 3

  Set the SIME and relevant Multi-Function interrupt enable bit 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 SIM 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 either a matching slave address, 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

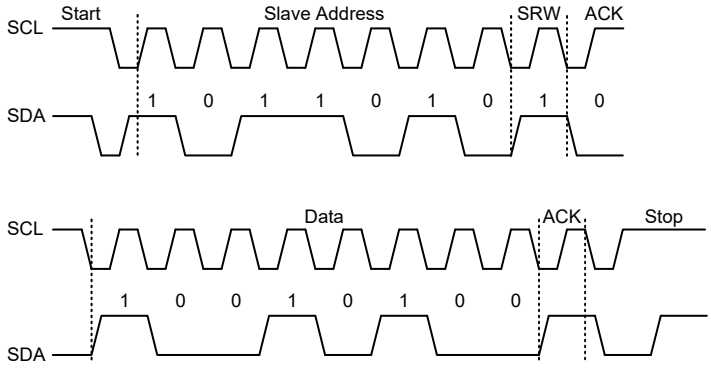
After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal 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 cleared to “0”.

I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 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.

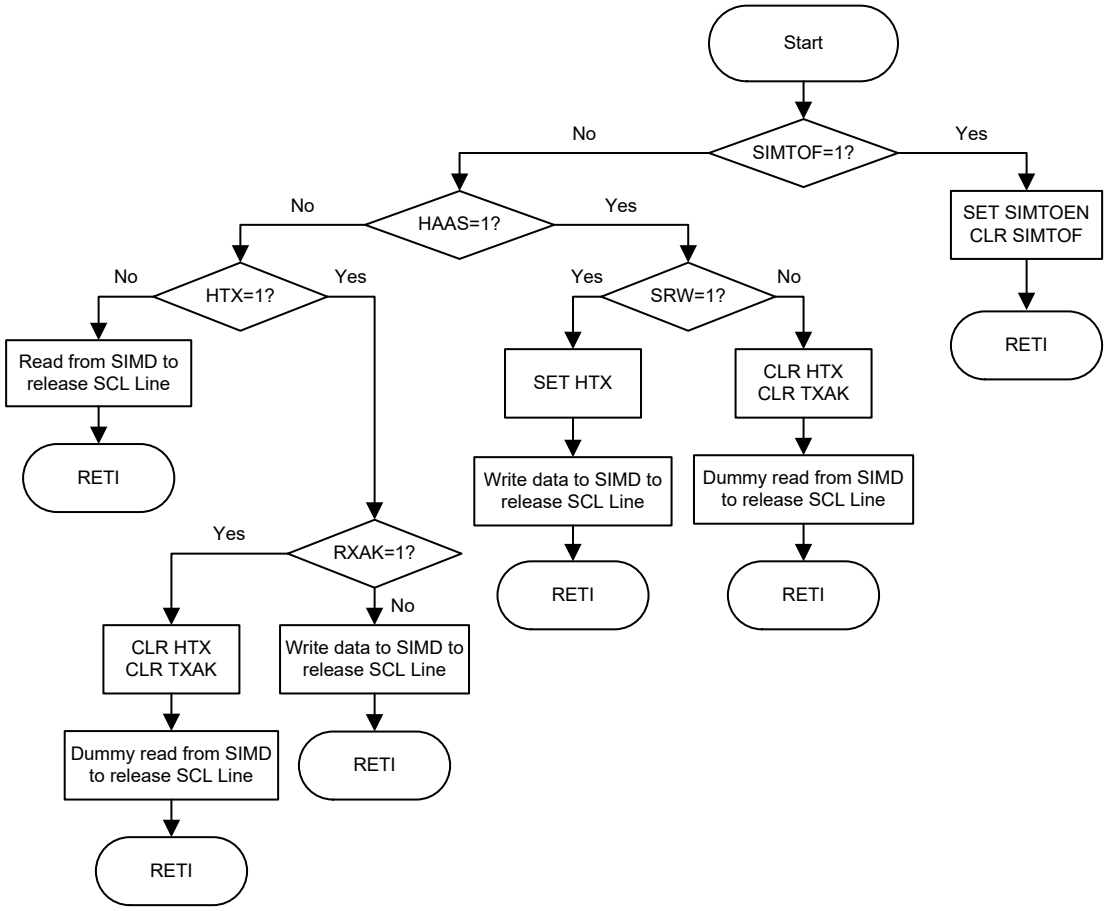


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

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.

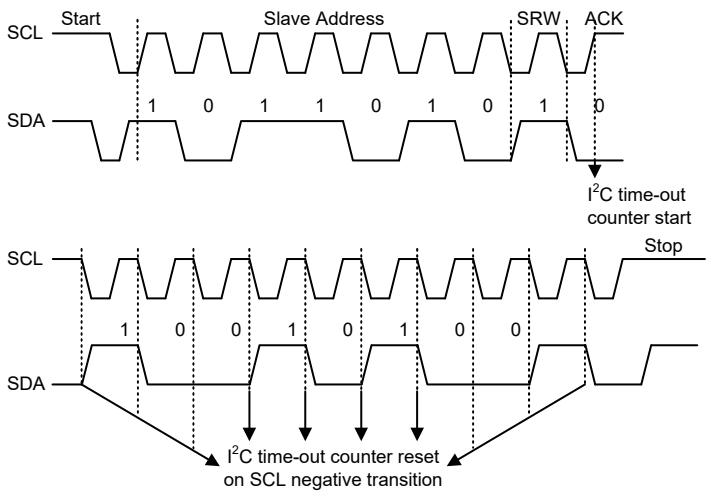
I²C Communication Timing Diagram



I²C Bus ISR Flow Chart

I²C Time-out Control

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



I²C Time-out

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 a SIM I²C interrupt which uses a multi-function 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 period selections which can be selected using the SIMTOS5~SIMTOS0 bits in the SIMTOC register. The time-out duration is calculated by the formula: ((1~64)×(32/f_{SUB})). This gives a time-out period which ranges from about 1ms to 64ms.

- **SIMTOC Register**

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

Bit 7 **SIMTOEN**: SIM I²C Time-out control
0: Disable
1: Enable

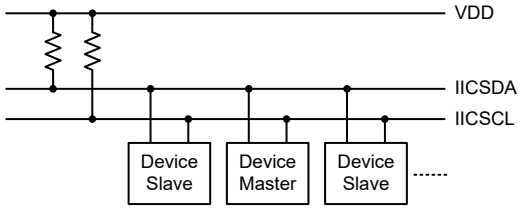
Bit 6 **SIMTOF**: SIM I²C Time-out flag
0: No time-out occurred
1: Time-out occurred
This bit is set high when time-out occurs and can only be cleared by application program.

Bit 5~0 **SIMTOS5~SIMTOS0**: SIM I²C Time-out period selection
I²C Time-out clock source is f_{SUB}/32.
I²C Time-out period is equal to (SIMTOS[5:0]+1)×(32/f_{SUB}).

Independent I²C Interface

The device contains an independent I²C function. It is important not to confuse this independent I²C function with the additional one contained within the combined SIM function, which is described in another section of this datasheet. This independent I²C function will carry with different pin names and register/bit names to distinguish it from the other one in the SIM.

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

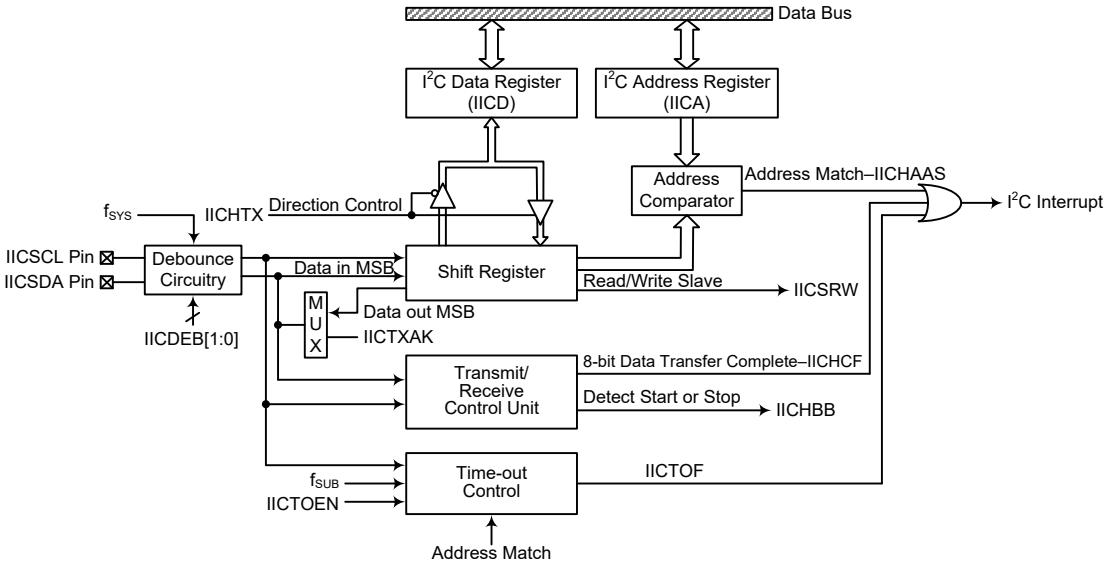
I²C Interface Operation

The I²C serial interface is a two-line interface, a serial data line, IICSDA, and serial clock line, IICSCL. 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 the device on the I²C bus is identified by a unique address which will be transmitted and received on the I²C bus.

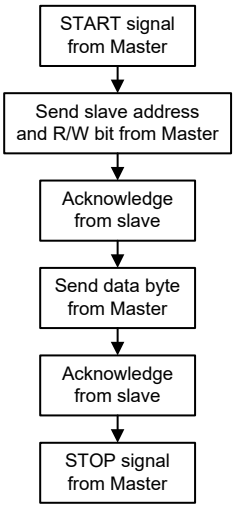
When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data; however, it is the master device that has overall control of the bus. For the device, which only operates in slave 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 IICSCL/IICSDA pin is still applicable even if the I²C device is activated and the related internal pull-high function could be controlled by its corresponding pull-high control register.

It is suggested that the device should not enter the IDLE/SLEEP mode during the I²C communication.



I²C Interface Block Diagram



I²C Interface Operation

The IICDEB1 and IICDEB0 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, IICC0, IICC1 and IICTOC, one address register IICA and one data register, IICD.

Register Name	Bit							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
IICC1	IICHCF	IICHAAS	IICHBB	IICHTX	IICTXAK	IICSRW	IICIAMWU	IICRXAK
IICD	D7	D6	D5	D4	D3	D2	D1	D0
IICA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
IICTOC	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0

I²C Register List

I²C Data Register

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

- **IICD 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**: I²C data register bit 7 ~ bit 0

I²C Address Register

The IICA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the IICA 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 IICA register, the slave device will be selected.

- **IICA Register**

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
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 7~1 **IICA6~IICA0**: I²C slave address
IICA6~IICA0 is the I²C slave address bit 6 ~ bit 0.
Bit 0 Unimplemented, read as "0"

I²C Control Registers

There are three control registers for the I²C interface, IICC0, IICC1 and IICTOC. The IICC0 register is used to control the enable/disable function and select the debounce time. The IICC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, IICTOC, is used to control the I²C time-out function and is described in the corresponding section.

• **IICC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **IICDEB1~IICDEB0**: I²C debounce time selection
00: No debounce
01: 2 system clock debounce
1x: 4 system clock debounce
Note that the I²C debounce circuit will operate normally if the system clock, f_{SYS}, is derived from the f_H clock or the IICIAMWU bit is equal to 0. Otherwise, the debounce circuit will have no effect and be bypassed.

Bit 1 **IICEN**: I²C enable control
0: Disable
1: Enable
The bit is the overall on/off control for the I²C interface. When the IICEN bit is cleared to zero to disable the I²C interface, the IICSDA and IICSCL lines will lose their I²C function and the I²C operating current will be reduced to a minimum value. When the bit is high the I²C interface is enabled. If the IICEN bit changes from low to high, the contents of the I²C control bits such as IICHTX and IICTXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as IICHCF, IICHAAS, IICHBB, IICSRW and IICRXAK will be set to their default states.

Bit 0 Unimplemented, read as "0"

• **IICC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	IICHCF	IICHAAS	IICHBB	IICHTX	IICTXAK	IICSRW	IICIAMWU	IICRXAK
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **IICHCF**: I²C bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer
The IICHCF 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. The following is an example of the flow of a two-byte I²C data transfer.
First, I²C slave device receives a start signal from I²C master and then IICHCF bit is automatically cleared to zero. Second, I²C slave device finishes receiving the 1st data byte and then IICHCF bit is automatically set high. Third, user read the 1st data byte from IICD register by the application program and then IICHCF bit is automatically cleared to zero. Fourth, I²C slave device finishes receiving the 2nd data byte and then IICHCF bit is automatically set to one and so on. Finally, I²C slave device receives a stop signal from I²C master and then IICHCF bit is automatically set high.

Bit 6 **IICHAAS**: I²C bus address match flag
0: Not address match
1: Address match
The IICHAAS 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 **IICHBB**: I²C bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy
The IICHBB 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 **IICHTX**: I²C slave device is transmitter or receiver selection
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **IICTXAK**: I²C bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag
The IICTXAK 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 IICTXAK bit to "0" before further data is received.

Bit 2 **IICSRW**: I²C slave read/write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode
The IICSRW 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 matches the slave address, which is when the IICHAAS flag is set high, the slave device will check the IICSRW flag to determine whether it should be in transmit mode or receive mode. If the IICSRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the IICSRW 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 **IICIAMWU**: I²C address match wake-up control
0: Disable
1: Enable
This bit should be set to 1 to enable the I²C address match wake-up from the SLEEP or IDLE Mode. If the IICIAMWU bit has been set high before entering either the SLEEP or IDLE mode to enable the I²C address match wake-up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

Bit 0 **IICRXAK**: I²C bus receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave does not receive acknowledge flag
The IICRXAK flag is the receiver acknowledge flag. When the IICRXAK flag is "0", it means that an 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 IICRXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the IICRXAK flag is "1". When this occurs, the slave transmitter will release the IICSDA 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 IICHAAS bit in the IICC1 register will be set and an I²C bus interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and IICTOF 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 IICSRW 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

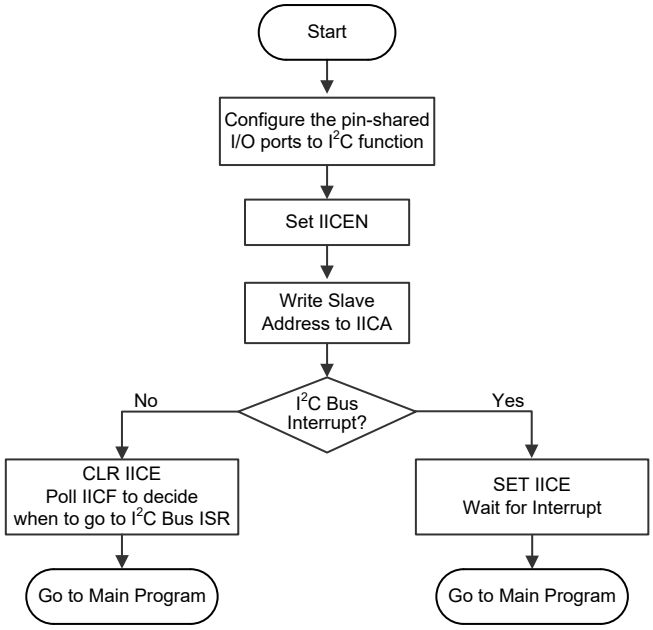
  Configure the corresponding pin-shared function as the I²C function pins and set the IICEN bit to "1" to enable the I²C bus.

- Step 2

  Write the slave address of the device to the I²C bus address register IICA.

- Step 3

  Set the I²C interrupt enable bit IICE in the interrupt control register to enable the I²C 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 IICHBB bit will be set. A START condition occurs when a high to low transition on the IICSDA line takes place when the IICSCL 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 slve 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 IICSRW bit of the IICC1 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 IICHAAS when the addresses match.

As an I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the IICHAAS and IICTOF bits should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer or from the 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 IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the IICSCL line.

I²C Bus Read/Write Signal

The IICSRW bit in the IICC1 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 IICSRW 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 IICSRW 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.

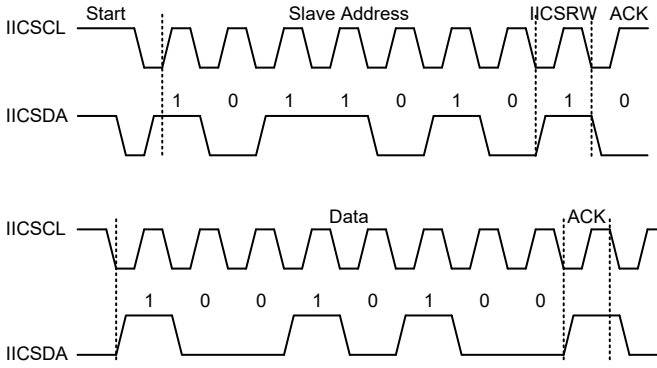
I²C Bus Slave Address Acknowledge Signal

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 IICHAAS flag is high, the addresses have matched and the slave device must check the IICSRW flag to determine if it is to be a transmitter or a receiver. If the IICSRW flag is high, the slave device should be setup to be a transmitter so the IICHTX bit in the IICC1 register should be set to "1". If the IICSRW flag is low, then the microcontroller slave device should be setup as a receiver and the IICHTX bit in the IICC1 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 IICSDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the IICD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the IICD register. If setup as a receiver, the slave device must read the transmitted data from the IICD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as IICTXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the IICRXAK bit in the IICC1 register to determine if it is to send another data byte, if not then it will release the IICSDA line and await the receipt of a STOP signal from the master.

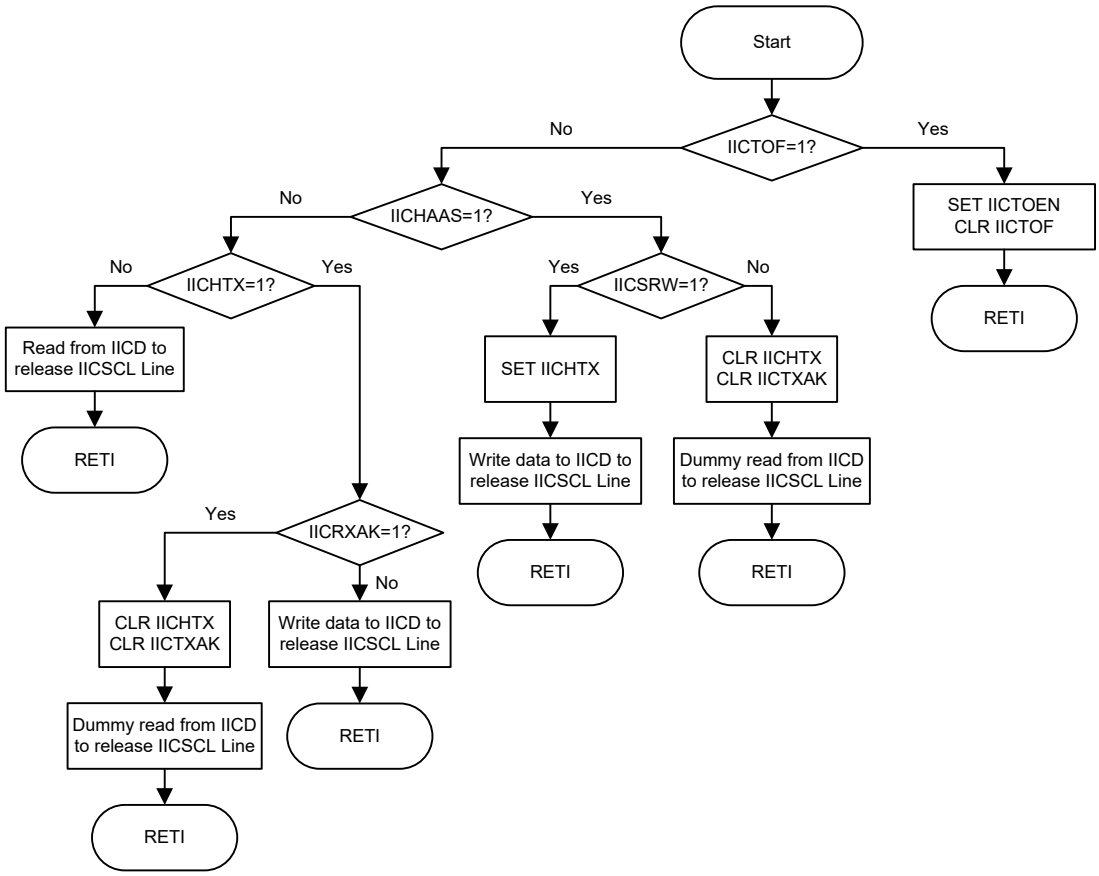


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=IICSRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (IICRXAK bit for transmitter, IICTXAK bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	.....	S	SA	SR	M	D	A	D	A	.....	P

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

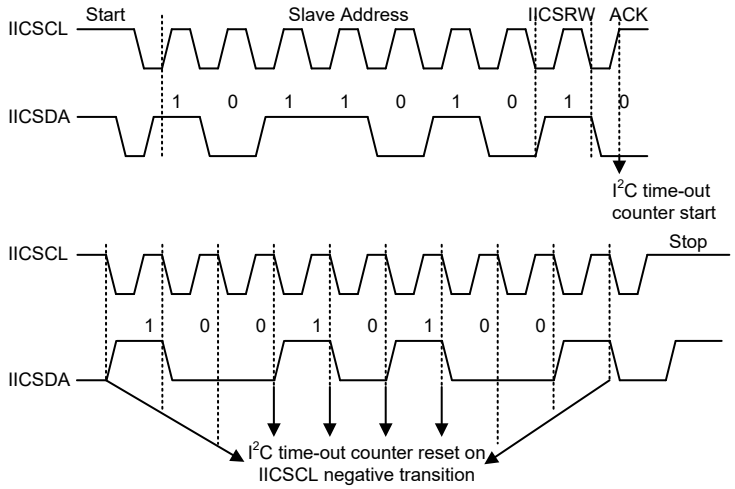
I²C Communication Timing Diagram



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 IICSCL falling edge. Before the next IICSCL falling edge arrives, if the time elapsed is greater than the time-out setup by the IICTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.



I²C Time-out

When an I²C time-out counter overflow occurs, the counter will stop and the IICTOEN bit will be cleared to zero and the IICTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an I²C 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
IICD, IICA, IICC0	No change
IICC1	Reset to POR condition

I²C Registers after Time-out

The IICTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using IICTOS5~IICTOS0 bits in the IICTOC register. The time-out time is given by the formula: $((1\sim64)\times(32/f_{SUB}))$. This gives a time-out period which ranges from about 1ms to 64ms.

- **IICTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0
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 **IICTOEN**: I²C time-out control
0: Disable
1: Enable

Bit 6 **IICTOF**: I²C time-out flag
0: No time-out occurred
1: Time-out occurred

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

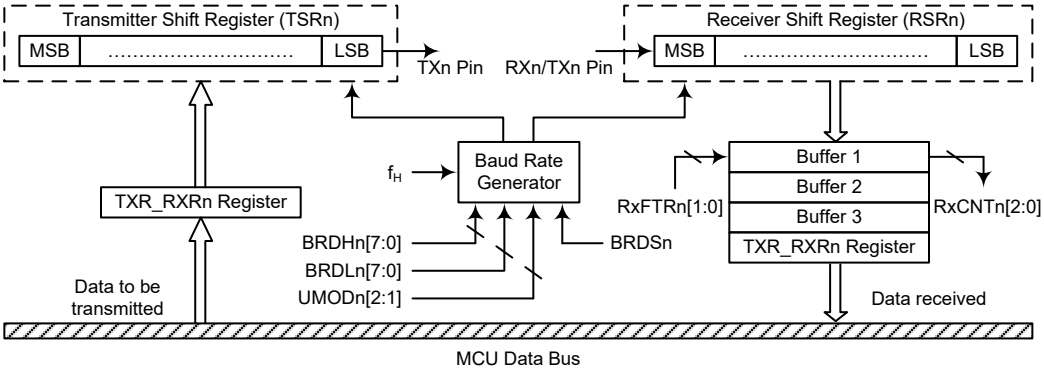
Bit 5~0 **IICTOS5~IICTOS0**: I^2C time-out period selection
I^2C time-out clock source is $f_{SUB}/32$.
I^2C time-out time is equal to (IICTOS[5:0]+1)×($32/f_{SUB}$).

UART Interfaces

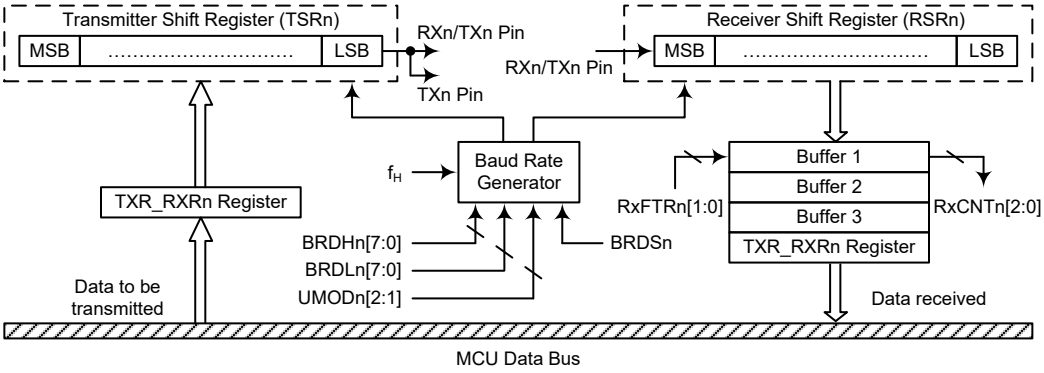
The device contains three integrated full-duplex or half-duplex asynchronous serial communications UART interfaces that enable communication with external devices that contain a serial interface. Each UART function has many features and can transmit and receive data serially by transferring a frame of 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=0~2)



UARTn Data Transfer Block Diagram – SWMn=1 (n=0~2)

UART External Pins

To communicate with an external serial interface, the internal UARTn has two external pins known as TXn and RXn/TXn, which are pin-shared with I/O or other pin functions. The TXn and RXn/TXn pin function should first be selected by the corresponding pin-shared function selection register before the UARTn function is used. Along with the UARTENn bit, the TXENn and RXENn bits, if set, will setup these pins to transmitter output and receiver input conditions. At this time the internal pull-high resistor related to the transmitter output pin will be disabled, while the internal pull-high resistor related to the receiver input pin is controlled by the corresponding I/O pull-high function control bit. When the 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, TXR_RXRn, in the Data Memory.

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 registers 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	TXRXn7	TXRXn6	TXRXn5	TXRXn4	TXRXn3	TXRXn2	TXRXn1	TXRXn0
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=0~2)

• **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 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 by a software sequence which will involve a read to the status register UnSR followed by an access to the TXR_RXRn data register.

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

The FERRn flag is the framing error flag. When this read only flag is "0", it indicates that there is no framing error. When the flag is "1", it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register UnSR followed by an access to the TXR_RXRn data register.

Bit 4 **OERRn**: Overrun error flag
0: No overrun error is detected
1: Overrun error is detected

The OERRn flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is "0", it indicates that there is no overrun error. When the flag is "1", it indicates that an overrun error occurs which will inhibit further transfers to the TXR_RXRn receive data register. The flag is cleared by a software sequence, which is a read to the status register UnSR followed by an access to the TXR_RXRn data register.

Bit 3 **RIDLEn**: Receiver status
0: Data reception is in progress (Data being received)
1: No data reception is in progress (Receiver is idle)

The RIDLEn flag is the receiver status flag. When this read only flag is "0", it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is "1", it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLEn bit is "1" indicating that the UARTn receiver is idle and the RXn/TXn pin stays in logic high condition.

Bit 2 **RXIFn**: Receive TXR_RXRn data register status
0: TXR_RXRn data register is empty
1: TXR_RXRn data register has available data and 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 is cleared when the UnSR register is read with RXIFn set, followed by a read from the TXR_RXRn register, and if the TXR_RXRn register has no data available.

Bit 1 **TIDLEn**: Transmission idle
0: Data transmission is in progress (Data being transmitted)
1: No data transmission is in progress (Transmitter is idle)

The TIDLEn flag is known as the transmission complete flag. When this read only flag is "0", it indicates that a transmission is in progress. This flag will be set high when the TXIFn flag is "1" and when there is no transmit data or break character being transmitted. When TIDLEn is equal to "1", the TXn pin becomes idle with the pin state in logic high condition. The TIDLEn flag is cleared 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 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 register 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 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 SWMn 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 RxCNTn register will be cleared, while the TIDLEn, TXIFn and RIDLEn bits will be set. Other control bits in UnCR1, UnCR2, UnCR3, UFCRn, BRDHn and BRDLn registers will remain unaffected. If the UARTn is active and the UARTENn bit is cleared, 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 TXRXn7 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. Replace the most significant bit position with a parity bit.

Bit 4~3 **PRTn1~PRTn0**: Parity type selection bits
00: Even parity for parity generator
01: Odd parity for parity generator
10: Mark parity for parity generator
11: Space parity for parity generator

These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.

Bit 2 **TXBRKn**: Transmit break character
0: No break character is transmitted
1: Break characters transmit

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

Bit 1 **RX8n**: Receive data bit 8 for 9-bit data transfer format (read only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8n. The 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 two UARTn control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UARTn Transmitter and Receiver as well as enabling the various UARTn interrupt sources. The register also serves to control the receiver STOP bit number selection, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	TXENn	RXENn	STOPSn	ADDENn	WAKEn	RIEn	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/TXn pin will be controlled by the UARTn. Clearing the RXENn bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RXn/TXn pin will be set in a floating state.

Bit 5 **STOPSn**: Number of Stop bits selection for receiver
0: One stop bit format is used
1: Two stop bits format is used

This bit determines if one or two stop bits are to be used for receiver. When this bit is equal to "1", two stop bits are used. If this bit is equal to "0", then only one stop bit is used. Two stop bits are used for transmitter.

Bit 4 **ADDENn**: Address detect function enable control
0: Address detect function is disabled
1: Address detect function is enabled

The bit named ADDENn is the address detect function enable control bit. When this bit is equal to "1", the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXRXn7 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	TXRXn7	TXRXn6	TXRXn5	TXRXn4	TXRXn3	TXRXn2	TXRXn1	TXRXn0
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 **TXRXn7~TXRXn0**: UARTn Transmit/Receive Data bit 7 ~ bit 0

• **BRDHn Register**

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

Bit 7~0 **D7~D0**: Baud rate divider high byte

The baud rate divider BRDn (BRDHn/BRDLn) defines the UARTn clock divider ratio.

Baud Rate=f_H/(BRDn+UMODn/8)

BRDn=16~65535 or 8~65535 depending on BRDSn

Note: 1. The BRDn value should not be set to less than 16 when BRDSn=0 or less than 8 when BRDSn=1, otherwise errors may occur.

2. The BRDLn must be written first and then BRDHn, otherwise errors may occur.

3. The BRDHn register should not be modified during data transmission process.

• **BRDLn Register**

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

Bit 7~0 **D7~D0**: Baud rate divider low byte

The baud rate divider BRDn (BRDHn/BRDLn) defines the UARTn clock divider ratio.

Baud Rate=f_H/(BRDn+UMODn/8)

BRDn=16~65535 or 8~65535 depending on BRDSn

Note: 1. The BRDn value should not be set to less than 16 when BRDSn=0 or less than 8 when BRDSn=1, otherwise errors may occur.

2. The BRDLn must be written first and then BRDHn, otherwise errors may occur.

3. The BRDLn register should not be modified during data transmission process.

• **UFCRn Register**

The UFCRn register is the FIFO control register which is used for UARTn modulation control, BRDn range selection and trigger level selection for RXIFn and interrupt.

Bit	7	6	5	4	3	2	1	0
Name	—	—	UMODn2	UMODn1	UMODn0	BRDSn	RxFTRn1	RxFTRn0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~3 **UMODn2~UMODn0**: UARTn Modulation Control bits

The modulation control bits are used to correct the baud rate of the received or transmitted UARTn signal. These bits determine if the extra UARTn clock cycle should be added in a UARTn bit time. The UMODn2~UMODn0 will be added to internal accumulator for every UARTn bit time. Until a carry to bit 3, the corresponding UARTn bit time increases a UARTn clock cycle.

Bit 2 **BRDSn**: BRDn range selection

0: BRDn range is from 16 to 65535
1: BRDn range is from 8 to 65535

The BRDSn is used to control the sampling point in a UARTn bit time. If the BRDSn bit 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 bit 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 received data bytes in the Receiver FIFO which is not read by the MCU. When Receiver FIFO receives one byte data, the RxCNTn will increase by one; when the MCU reads one byte data from the Receiver FIFO, the RxCNTn will decrease by one. If there are 4 bytes of data in the Receiver FIFO, the 5th data will be saved in the shift register. If there is 6th data, the 6th data will be saved in the shift register. But the RxCNTn remains the value of 4. The RxCNTn will be cleared when reset occurs or UARTENn=1. This register is read only.

Baud Rate Generator

To setup the speed of the serial data communication, the UARTn function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in 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.

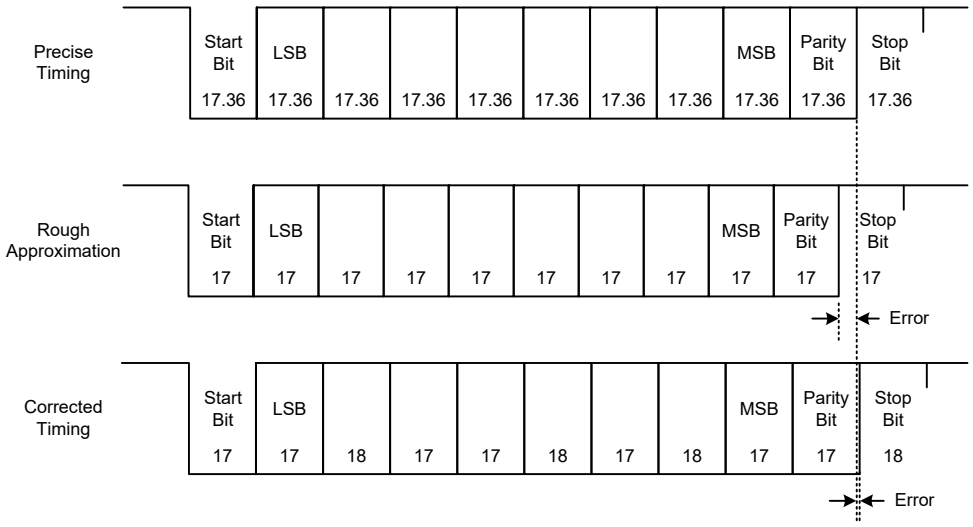
Fraction Addition	Carry to Bit 3	UARTn Bit Time Sequence	Extra UARTn Clock Cycle
0000b+0011b=0011b	No	Start bit	No
0011b+0011b=0110b	No	D0	No
0110b+0011b=1001b	Yes	D1	Yes
1001b+0011b=1100b	No	D2	No
1100b+0011b=1111b	No	D3	No
1111b+0011b=0010b	Yes	D4	Yes
0010b+0011b=0101b	No	D5	No
0101b+0011b=1000b	Yes	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, which is the setting at power-on. 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 and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UARTn function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UARTn will also reset the error and status flags with bits TXENn, RXENn, TXBRKn, RXIFn, OERRn, FERRn, PERRn and NFn as well as register RxCNTn being cleared while bits TIDLEn, TXIFn and RIDLEn will be set. The remaining control bits in the UnCR1, UnCR2, UnCR3, UFCRn, BRDHn and BRDLn registers will remain unaffected. If the UARTENn bit in the UnCR1 register is cleared while the UARTn is active, then all pending transmissions and receptions will be immediately suspended and the UARTn will be reset to a condition as defined above. If the UARTn is then subsequently re-enabled, it will restart again in the same configuration.

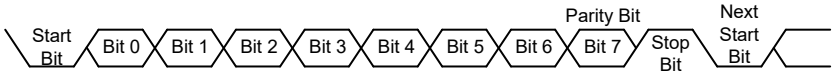
Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UnCR1 and UnCR2 registers. The 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 only configurable for the receiver. The transmitter uses two stop bits.

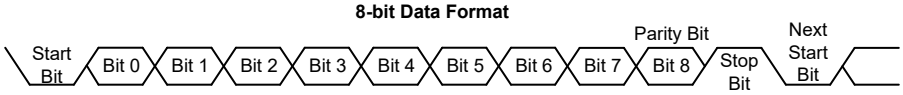
Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1 or 2
1	7	0	1	1 or 2
1	7	1	0	1 or 2
Example of 9-bit Data Formats				
1	9	0	0	1 or 2
1	8	0	1	1 or 2
1	8	1	0	1 or 2

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 TXn 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 TXn 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. A 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. A 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 break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13×N '0' bits and stop bits, where N=1, 2, etc. If a break character is to be transmitted then the 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 pin, LSB first. In the read mode, the TXR_RXRn register forms a buffer between the internal bus and the receiver shift register. The TXR_RXRn register is a four byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR_RXRn before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERRn will be subsequently indicated. For continuous multi-byte data transmission, it is strongly recommended that the receiver uses two stop bits to avoid a receiving error caused by the accumulated error of the receiver baud rate frequency.

The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNOn, PRTn1~PRTn0, PRENn and STOPSn bits to define the word length, parity type and number of stop bits.
- Setup the BRDHn and BRDLn registers and the UMODn2~UMODn0 bits to select the desired baud rate.
- Set the RXENn bit to ensure that the RXn/TXn 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 the TXR_RXRn register has data available, the number of the available data bytes can be checked by polling the RxCNTn 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 RIEn bit is set, then an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

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

1. A 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 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 this fifth byte has been entirely shifted in, the data should be read from the TXR_RXRn register. If this is not done, the overrun error flag OERRn will be consequently indicated.

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

- The OERRn flag in the UnSR register will be set.
- The TXR_RXRn contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIEn bit is set.

When the OERRn flag is set to "1", it is necessary to read five data bytes from the four-byte deep receiver FIFO and the shift register immediately to avoid unexpected errors, such as the UARTn is unable to receive data. If such an error occurs, clear the RXENn bit to "0" then set it to "1" again to continue data reception.

The OERRn flag can be cleared by an access to the 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.

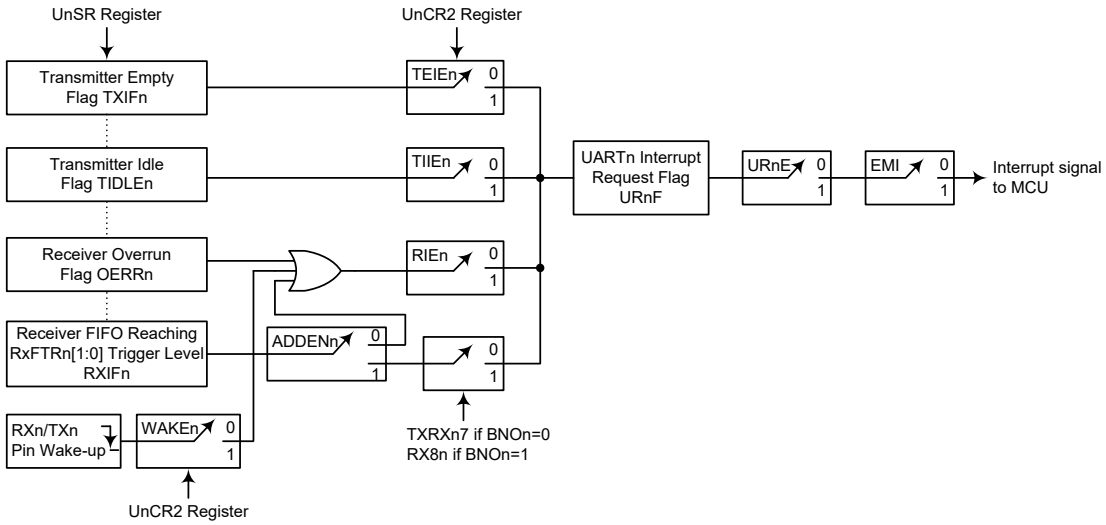
UART Interrupt Structure

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 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=0~2)

Address Detect Mode

Setting the Address Detect Mode bit, ADDENn, in the UnCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the 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 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=0~2)

UART Power Down and Wake-up

When the UARTn clock (f_H) is off, the UARTn will cease to function, all clock sources to the module are shutdown. If the UARTn clock (f_H) is off 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 enters the IDLE or SLEEP mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the UnSR, UnCR1, UnCR2, UnCR3, 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 UARTn clock (f_H) is off, then a falling edge on the RXn/TXn pin will trigger 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 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.

Touch Key Function

The device provides multiple touch key functions. The touch key function is fully integrated and requires no external components, allowing touch key functions to be implemented by the simple manipulation of internal registers.

Touch Key Structure

The touch keys are pin-shared with the I/O pins, with the desired function chosen via the corresponding selection register bits. Keys are organised into several groups, with each group known as a module and having a module number, M0 to M5. Each module is a fully independent set of four Touch Keys and each Touch Key has its own oscillator. Each module contains its own control logic circuits and register set. Examination of the register names will reveal the module number it is referring to.

Total Key Number	Touch Key Module		Shared I/O Pin
24	Mn (n=0~5)	M0	KEY1~KEY4
		M1	KEY5~KEY8
		M2	KEY9~KEY12
		M3	KEY13~KEY16
		M4	KEY17~KEY20
		M5	KEY21~KEY24

Touch Key Structure

Touch Key Register Definition

Each touch key module, which contains four touch key functions, has its own suite registers. The following table shows the register set for the touch key module. The Mn within the register name refers to the Touch Key module number. The device has six Touch Key Modules.

Register Name	Description
TKTMR	Touch key time slot 8-bit counter preload register
TKC0	Touch key function control register 0
TKC1	Touch key function control register 1
TK16DL	Touch key function 16-bit counter low byte
TK16DH	Touch key function 16-bit counter high byte
TKMn16DL	Touch key module n 16-bit C/F counter low byte
TKMn16DH	Touch key module n 16-bit C/F counter high byte
TKMnROL	Touch key module n reference oscillator capacitor selection low byte
TKMnROH	Touch key module n reference oscillator capacitor selection high byte
TKMnC0	Touch key module n control register 0
TKMnC1	Touch key module n control register 1
TKMnC2	Touch key module n control register 2

Touch Key Function Register Definition (n=0~5)

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	TKRAMC	TKRCOV	TKST	TKCFOV	TK16OV	—	TKMOD	TKBUSY
TKC1	D7	D6	D5	TSCS	TK16S1	TK16S0	TKFS1	TKFS0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKMn16DL	D7	D6	D5	D4	D3	D2	D1	D0
TKMn16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKMnROL	D7	D6	D5	D4	D3	D2	D1	D0
TKMnROH	—	—	—	—	—	—	D9	D8
TKMnC0	—	—	MnDFEN	MnFILEN	MnSOFC	MnSOF2	MnSOF1	MnSOF0
TKMnC1	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN
TKMnC2	MnSK31	MnSK30	MnSK21	MnSK20	MnSK11	MnSK10	MnSK01	MnSK00

Touch Key Function Register List (n=0~5)

- **TKTMR 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**: Touch key time slot 8-bit counter preload register

The touch key time slot counter preload register is used to determine the touch key time slot overflow time. The time slot unit period is obtained by a 5-bit counter and equal to 32 time slot clock cycles. Therefore, the time slot counter overflow time is equal to the following equation shown.

Time slot counter overflow time=(256-TKTMR[7:0])×32t_{TSC}, where t_{TSC} is the time slot counter clock period.

- **TKC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	TKRAMC	TKRCOV	TKST	TKCFOV	TK16OV	—	TKMOD	TKBUSY
R/W	R/W	R/W	R/W	R/W	R/W	—	R/W	R
POR	0	0	0	0	0	—	0	0

Bit 7 **TKRAMC**: Touch key data memory access control

0: Accessed by MCU

1: Accessed by touch key module

This bit determines that the touch key data memory is used by the MCU or the touch key module. However, the touch key module will have the priority to access the touch key data memory when the touch key module operates in the auto scan mode, i.e., the TKST bit state is changed from 0 to 1 when the TKMOD bit is cleared to 0. After the touch key auto scan operation is completed, i.e., the TKBUSY bit state is changed from 1 to 0, the touch key data memory access will be controlled by the TKRAMC bit. Therefore, it is recommended to set the TKRAMC bit to 1 when the touch key module operates in the auto scan mode. Otherwise, the contents of the touch key data memory may be modified as this data memory space is configured by the touch key module followed by the MCU access. When the TKRAMC bit is set to 1, MCU reading the touch key data memory will obtain uncertain values as the touch key data memory is controlled by the touch key circuit.

Bit 6 **TKRCOV**: Touch key time slot counter overflow flag

0: No overflow occurs

1: Overflow occurs

This bit can be accessed by application program. When this bit is set by touch key time slot counter overflow, the corresponding touch key interrupt request flag will be set. However, if this bit is set by application program, the touch key interrupt request flag will not be affected. Therefore, this bit can not be set by application program but must be cleared to 0 by application program.

In the auto scan mode, if module 0 or all module time slot counter, selected by TSCS bit, overflows but the touch key auto scan operation is not completed, the TKRCOV bit will not be set. At this time, all module touch key module 16-bit C/F counter, touch key function 16-bit counter and 5-bit time slot unit period counter will be automatically cleared but the 8-bit time slot counter will be reloaded from the 8-bit time slot counter preload register. When the touch key auto scan operation is completed, the TKRCOV bit and the Touch Key Module Interrupt request flag, TKMF, will be set and all module keys and reference oscillators will automatically stop. All touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off.

In the manual scan mode, if module 0 or all module time slot counter, selected by TSCS bit, overflows, the TKRCOV bit and the Touch Key Module Interrupt request flag, TKMF, will be set and all module keys and reference oscillators will automatically stop. All touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off.

Bit 5 **TKST**: Touch key detection start control
0: Stopped or no operation
0→1: Start detection

In all modules, the touch key module 16-bit C/F counter, touch key function 16-bit counter and 5-bit time slot unit period counter will automatically be cleared when this bit is cleared to zero. However, the 8-bit programmable time slot counter will not be cleared. When this bit is changed from low to high, the touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be switched on together with the key and reference oscillators to drive the corresponding counters.

Bit 4 **TKCFOV**: Touch key module 16-bit C/F counter overflow flag
0: No overflow occurs
1: Overflow occurs

This bit is set by touch key module 16-bit C/F counter overflow and must be cleared to 0 by application program.

Bit 3 **TK16OV**: Touch key function 16-bit counter overflow flag
0: No overflow occurs
1: Overflow occurs

This bit is set by touch key function 16-bit counter overflow and must be cleared to 0 by application program.

Bit 2 Unimplemented, read as “0”

Bit 1 **TKMOD**: Touch key scan mode selection
0: Auto scan mode
1: Manual mode

In the manual scan mode the reference oscillator capacitor value should be properly configured before the scan operation begins and the touch key module 16-bit C/F counter value should be read by application program after the scan operation finishes.

In the auto scan mode the data movement which is described above is implemented by hardware. The individual reference oscillator capacitor value and 16-bit C/F counter content for all scanned keys will be read from and written into a dedicated Touch Key Data Memory area. In the auto scan mode the keys to be scanned can be arranged in a specific sequence which is determined by the MnSK3[1:0]~MnSK0[1:0] bits in the TKMnC2 register. The scan operation will not be stopped until all arranged keys are scanned.

Bit 0 **TKBUSY**: Touch key scan operation busy flag
0: Not busy – No scan operation is executed or scan operation is completed
1: Busy – Scan operation is executing

This bit indicates whether the touch key scan operation is executing or not. It is set to 1 when the TKST bit is set high to start the scan operation.

In the manual scan mode this bit is cleared to 0 automatically when module 0 or all module time slot counter, selected by TSCS bit, overflows. In the auto scan mode this bit is cleared to 0 automatically when the touch key scan operation is completed.

• **TKC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	TSCS	TK16S1	TK16S0	TKFS1	TKFS0
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	1	1

Bit 7~5 **D7~D5**: Data bits for test only
These bits are used for test purpose only and must be kept as "000" for normal operations.

Bit 4 **TSCS**: Touch Key modules time slot counter selection
0: Each module use its own time slot counter
1: All touch key modules use module 0 time slot counter

Bit 3~2 **TK16S1~TK16S0**: Touch key function 16-bit counter clock source selection
00: f_{SYS}
01: $f_{SYS}/2$
10: $f_{SYS}/4$
11: $f_{SYS}/8$

Bit 1~0 **TKFS1~TKFS0**: Touch key oscillator and Reference oscillator frequency selection
00: 1MHz
01: 2.5MHz
10: 7.5MHz
11: 11MHz

• **TK16DH/TK16DL – Touch Key Function 16-bit Counter Register Pair**

Register	TK16DH								TK16DL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register pair is used to store the touch key function 16-bit counter value. This 16-bit counter can be used to calibrate the reference or key oscillator frequency. When the touch key time slot counter overflows in the manual scan mode, this 16-bit counter will be stopped and the counter content will be unchanged. However, this 16-bit counter content will be cleared to zero at the end of the time slot 0, slot 1 and slot 2 but kept unchanged at the end of the time slot 3 in the auto scan mode. This register pair will be cleared to zero when the TKST bit is cleared to zero.

• **TKMn16DH/TKMn16DL – Touch Key Module n 16-bit C/F Counter Register Pair**

Register	TKMn16DH								TKMn16DL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register pair is used to store the touch key module n 16-bit C/F counter value. This 16-bit C/F counter will be stopped and the counter content will be kept unchanged when the touch key time slot counter overflows in the manual scan mode. However, this 16-bit C/F counter content will be cleared to zero at the end of the time slot 0, slot 1 and slot 2 after it is written to the touch key data memory but kept unchanged at the end of the time slot 3 in the auto scan mode. This register pair will be cleared to zero when the TKST bit is cleared to zero.

• **TKMnROH/TKMnROL – Touch Key Module n Reference Oscillator Capacitor Selection Register Pair**

Register	TKMnROH								TKMnROL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	—	—	—	—	—	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	—	—	—	0	0	0	0	0	0	0	0	0	0

This register pair is used to setup the touch key module n reference oscillator capacitor value. This register pair will be loaded with the corresponding next time slot capacitor value from the dedicated touch key data memory at the end of the current time slot when the auto scan mode is selected.

The reference oscillator internal capacitor value=(TKMnRO[9:0]×50pF)/1024

• **TKMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	MnDFEN	MnFILEN	MnSOFC	MnSOF2	MnSOF1	MnSOF0
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 **MnDFEN**: Touch key module n multi-frequency control
0: Disable
1: Enable
This bit is used to control the touch key oscillator frequency doubling function. When this bit is set to 1, the key oscillator frequency will be doubled.

Bit 4 **MnFILEN**: Touch key module n filter function control
0: Disable
1: Enable

Bit 3 **MnSOFC**: Touch key module n C/F oscillator frequency hopping function control selection
0: Controlled by the MnSOF2~MnSOF0 bits
1: Controlled by hardware circuit
This bit is used to select the touch key oscillator frequency hopping function control method. When this bit is set to 1, the key oscillator frequency hopping function is controlled by the hardware circuit regardless of the MnSOF2~MnSOF0 bits value.

Bit 2~0 **MnSOF2~MnSOF0**: Touch key module n Reference and Key oscillators hopping frequency select
000: 1.020MHz
001: 1.040MHz
010: 1.059MHz
011: 1.074MHz
100: 1.085MHz
101: 1.099MHz
110: 1.111MHz
111: 1.125MHz
These bits are used to select the touch key oscillator frequency for the hopping function. Note that these bits are only available when the MnSOFC bit is cleared to 0.
The frequencies mentioned here are only for the condition where the key and reference oscillator frequency is selected to be 1MHz, these values will be changed when the external or internal capacitor has different values. Users can adjust the key and reference oscillator frequency in scale when any other frequency is selected.

• **TKMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN
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 **MnTSS**: Touch key module n time slot counter clock source selection
0: Touch key module n reference oscillator
1: $f_{SYS}/4$

Bit 6 Unimplemented, read as "0"

Bit 5 **MnROEN**: Touch key module n Reference oscillator enable control
0: Disable
1: Enable

In the manual scan mode, this bit is used to enable/disable the touch key module n reference oscillator. The module n reference oscillator should first be enabled by setting the MnROEN bit high before setting the TKST bit from low to high if the reference oscillator is selected to be used and will be automatically disabled when the TKBUSY bit is changed from high to low.

In the auto scan mode, this bit is controlled by hardware automatically. For the module 0, when the TKST bit changes from low to high, the M0ROEN bit will be set high automatically. For the other module n (n≠0), if the condition that MnK4EN~MnK1EN≠0000B, MnTSS=0 and TSCS=0 is satisfied, the MnROEN bit will be set high automatically when the TKST bit changes from low to high, while in other conditions the MnROEN bit will not be affected by the TKST bit setting. When the TKBUSY bit is changed from high to low, the MnROEN bit will automatically be cleared to zero to disable the reference oscillator.

Bit 4 **MnKOEN**: Touch key module n Key oscillator enable control
0: Disable
1: Enable

In the manual scan mode, this bit is used to enable/disable the touch key module n key oscillator. The touch key module n key oscillator should first be enabled by setting the MnKOEN bit high before setting the TKST bit from low to high if the relevant key is enabled to be scanned and will be disabled automatically when the TKBUSY bit is changed from high to low.

In the auto scan mode, the MnKOEN bit will be set high automatically to enable the key oscillator when the TKST bit changes from low to high. When the TKBUSY bit is changed from high to low, the MnKOEN bit will automatically be cleared to zero to disable the key oscillator.

Bit 3 **MnK4EN**: Touch key module n Key4 enable control

MnK4EN	Touch Key Module n – Mn					
	M0	M1	M2	M3	M4	M5
0: Disable	I/O or other function					
1: Enable	KEY4	KEY8	KEY12	KEY16	KEY20	KEY24

Bit 2 **MnK3EN**: Touch key module n Key3 enable control

MnK3EN	Touch Key Module n – Mn					
	M0	M1	M2	M3	M4	M5
0: Disable	I/O or other function					
1: Enable	KEY3	KEY7	KEY11	KEY15	KEY19	KEY23

Bit 1 **MnK2EN**: Touch key module n Key2 enable control

MnK2EN	Touch Key Module n – Mn					
	M0	M1	M2	M3	M4	M5
0: Disable	I/O or other function					
1: Enable	KEY2	KEY6	KEY10	KEY14	KEY18	KEY22

Bit 0 **MnK1EN**: Touch key module n Key1 enable control

MnK1EN	Touch Key Module n – Mn					
	M0	M1	M2	M3	M4	M5
0: Disable	I/O or other functions					
1: Enable	KEY1	KEY5	KEY9	KEY13	KEY17	KEY21

• **TKMnC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	MnSK31	MnSK30	MnSK21	MnSK20	MnSK11	MnSK10	MnSK01	MnSK00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	1	0	0

Bit 7~6 **MnSK31~MnSK30**: Touch key module n time slot 3 key scan selection
00: Key1
01: Key2
10: Key3
11: Key4
These bits are used to select the desired scan key in time slot 3 and only available in the auto scan mode.

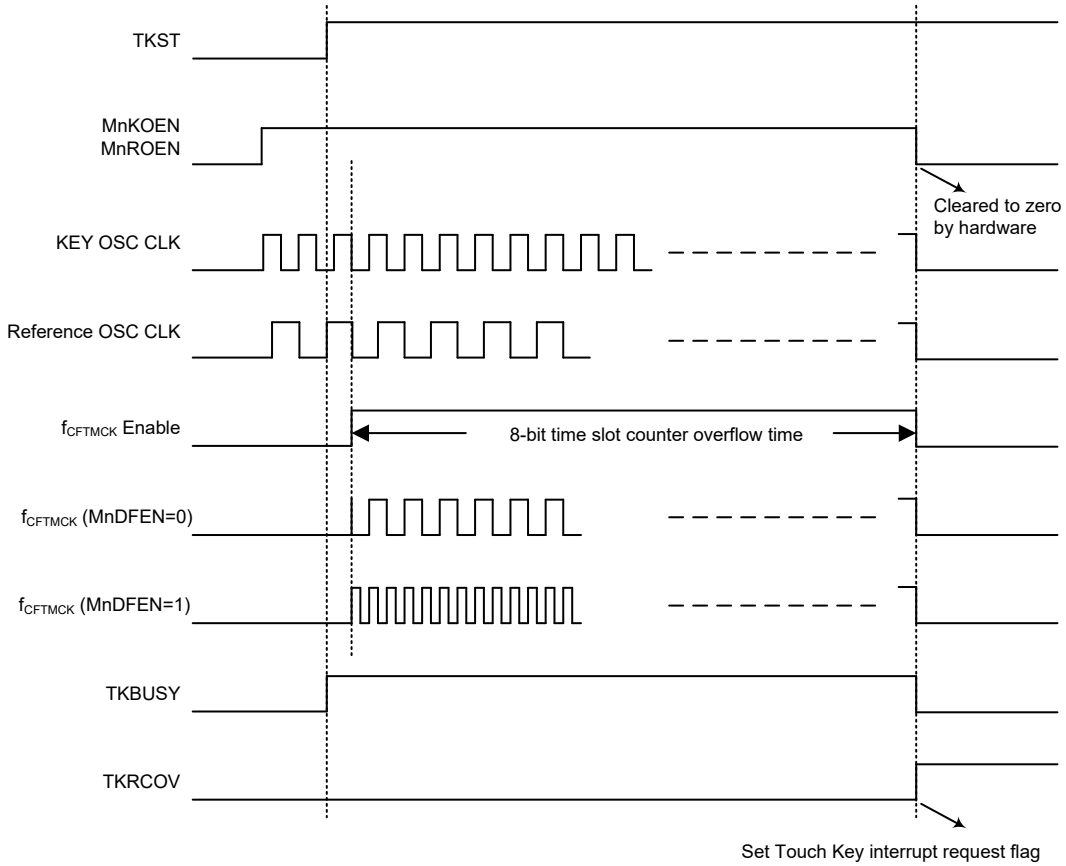
Bit 5~4 **MnSK21~MnSK20**: Touch key module n time slot 2 key scan selection
00: Key1
01: Key2
10: Key3
11: Key4
These bits are used to select the desired scan key in time slot 2 and only available in the auto scan mode.

Bit 3~2 **MnSK11~MnSK10**: Touch key module n time slot 1 key scan selection
00: Key1
01: Key2
10: Key3
11: Key4
These bits are used to select the desired scan key in time slot 1 and only available in the auto scan mode.

Bit 1~0 **MnSK01~MnSK00**: Touch key module n time slot 0 key scan selection
00: Key1
01: Key2
10: Key3
11: Key4
These bits are used to select the desired scan key in time slot 0 in the auto scan mode or used as the multiplexer for scan key selection in the manual mode.

Touch Key Operation

When a finger touches or is in proximity to a touch pad, the capacitance of the pad will increase. By using this capacitance variation to change slightly the frequency of the internal sense oscillator, touch actions can be sensed by measuring these frequency changes. Using an internal programmable divider the reference clock is used to generate a fixed time period. By counting a number of generated clock cycles from the sense oscillator during this fixed time period touch key actions can be determined.



Touch Key Manual Scan Mode Timing Diagram

Each touch key module contains four touch key inputs, which are shared with logical I/O pins, and the desired function is selected using the corresponding pin-shared control register bits. The touch key has its own independent sense oscillator. There are therefore four sense oscillators within a touch key module.

During this reference clock fixed interval, the number of clock cycles generated by the sense oscillator is measured, and it is this value that is used to determine if a touch action has been made or not. At the end of the fixed reference clock time interval a Touch Key interrupt signal will be generated in the manual scan mode.

Using the TSCS bit in the TKC1 register can select the module 0 time slot counter as the time slot counter for all modules. All modules use the same start signal, TKST, in the TKC0 register. The touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter in the module will be automatically cleared when the TKST bit is cleared to zero, but the 8-bit programmable time slot counter will not be cleared. The overflow time is setup by users. When the TKST bit changes from low to high, the 16-bit C/F counter, 16-bit counter, 5-bit time slot unit

period counter and 8-bit time slot timer counter will be automatically switched on.

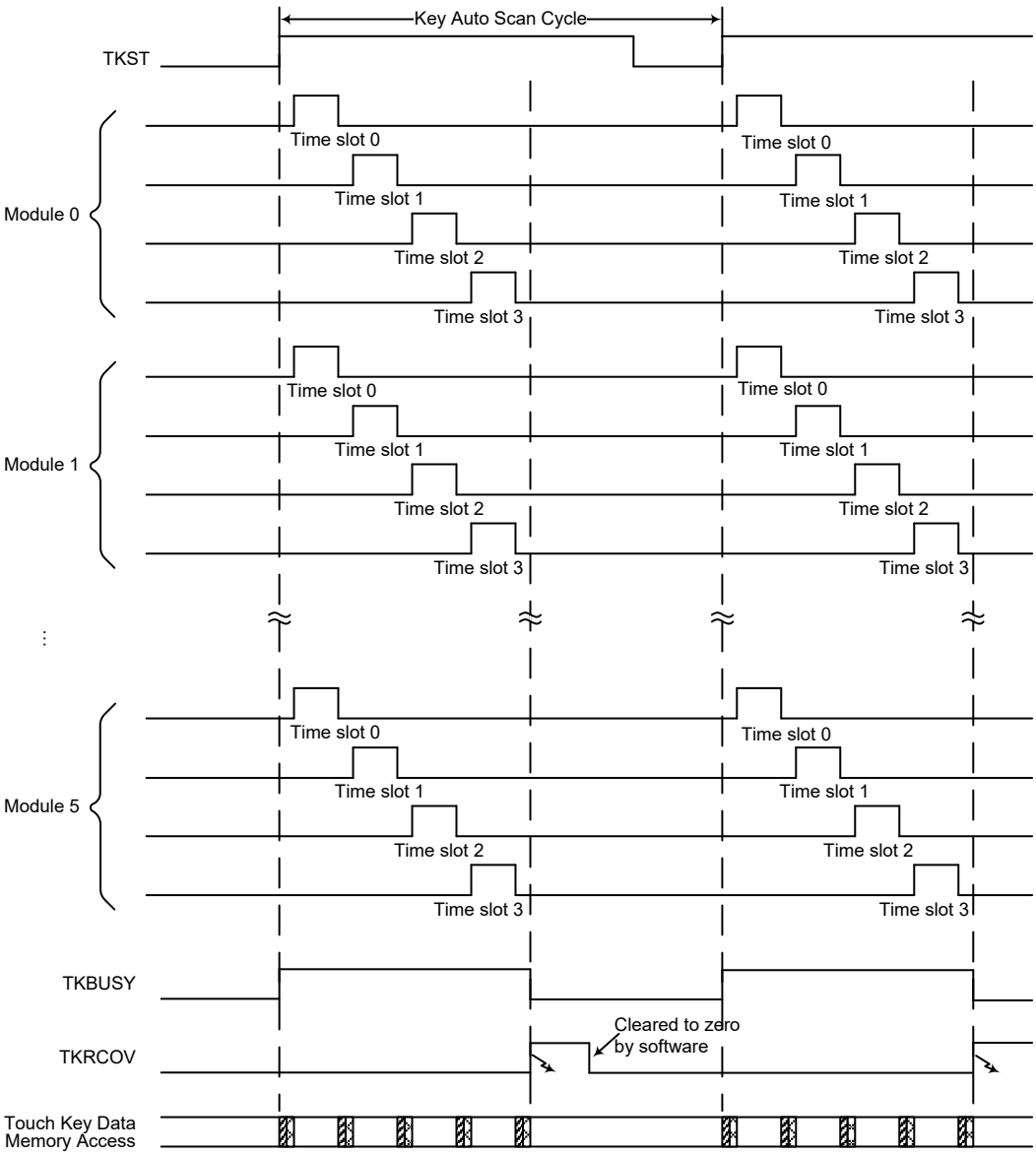
The key oscillator and reference oscillator in all modules will be automatically stopped and the 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched off when the time slot counter overflows. The clock source for the time slot counter is sourced from the module n reference oscillator or $f_{SYS}/4$ which is selected using the MnTSS bit in the TKMnC1 register. The reference oscillator and key oscillator will be enabled by setting the MnROEN bit and MnKOEN bits in the TKMnC1 register.

When the time slot counter in all the touch key modules or in the touch key module 0 overflows, an actual Touch Key interrupt will take place. The touch keys mentioned here are the keys which are enabled.

Each touch key module consists of four touch keys, KEY1~KEY4 are contained in module 0, KEY5~KEY8 are contained in module 1, KEY9~KEY12 are contained in module 2, KEY13~KEY16 are contained in module 3, KEY17~KEY20 are contained in module 4, KEY21~KEY24 are contained in module 5. Each touch key module has an identical structure.

Auto Scan Mode

There are two scan modes contained for the touch key function. The auto scan mode and the manual scan mode are selected using the TKMOD bit in the TKC0 register. The auto scan mode can minisize the load of the application program and improve the touch key scan operation performance. When the TKMOD bit is set to 0, the auto scan mode is selected to scan the module keys in a specific sequence determined by the MnSK3[1:0]~MnSK0[1:0] bits in the TKMnC2 register.



: Set Touch Key interrupt request flag

: Read 2N bytes from Touch Key Data Memory to TKMnROH/TKMnROL registers

: Write 2N bytes from TKMn16DH/TKMn16DL registers to Touch Key Data Memory

N = Touch Key Module Number; n = Module Serial Number

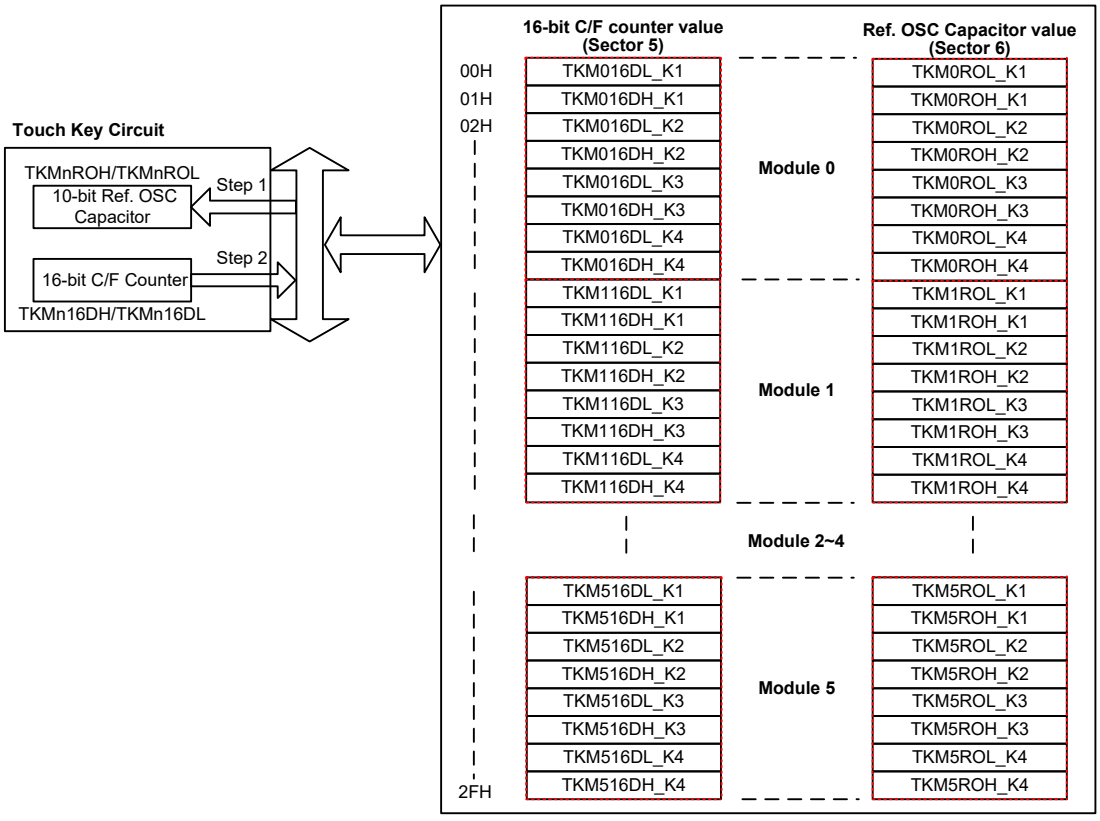
Touch Key Auto Scan Mode Timing Diagram

In the auto scan mode the module n key oscillator and reference oscillator which are required to be used will be enabled by hardware automatically when the TKST bit is set from low to high and disabled automatically when the TKBUSY bit changes from high to low. When the TKST bit is set from low to high in the auto scan mode, the internal capacitor value of the reference oscillator for the selected key to be scanned in the time slot 0 will first be read from a specific location of the dedicated touch key data memory and loaded into the corresponding TKMnROH/TKMnROL registers. Then the 16-bit C/F counter value will be written into the corresponding location of the

last time slot 3 scanned key in the touch key data memory. After this, the selected key will start to be scanned in time slot 0. At the end of the time slot 0 key scan operation, the reference oscillator internal capacitor value for the next selected key will be read from the touch key data memory and loaded into the next TKMnROH/TKMnROL registers. Then the 16-bit C/F counter value of the current scanned key will be written into the corresponding touch key data memory. The whole auto scan operation will sequentially be carried out in the above specific way from time slot 0 to time slot 3. At the end of the time slot 3 key scan operation, the reference oscillator internal capacitor value for the time slot 0 selected key will again be read from the touch key data memory and loaded into the corresponding TKMnROH/TKMnROL registers. Then the 16-bit C/F counter value will be written into the relevant location of the time slot 3 scanned key in the touch key data memory. After all selected keys are scanned, the TKRCOV bit will be set high and the TKBUSY bit will be cleared to zero as well as an auto scan mode operation is completed.

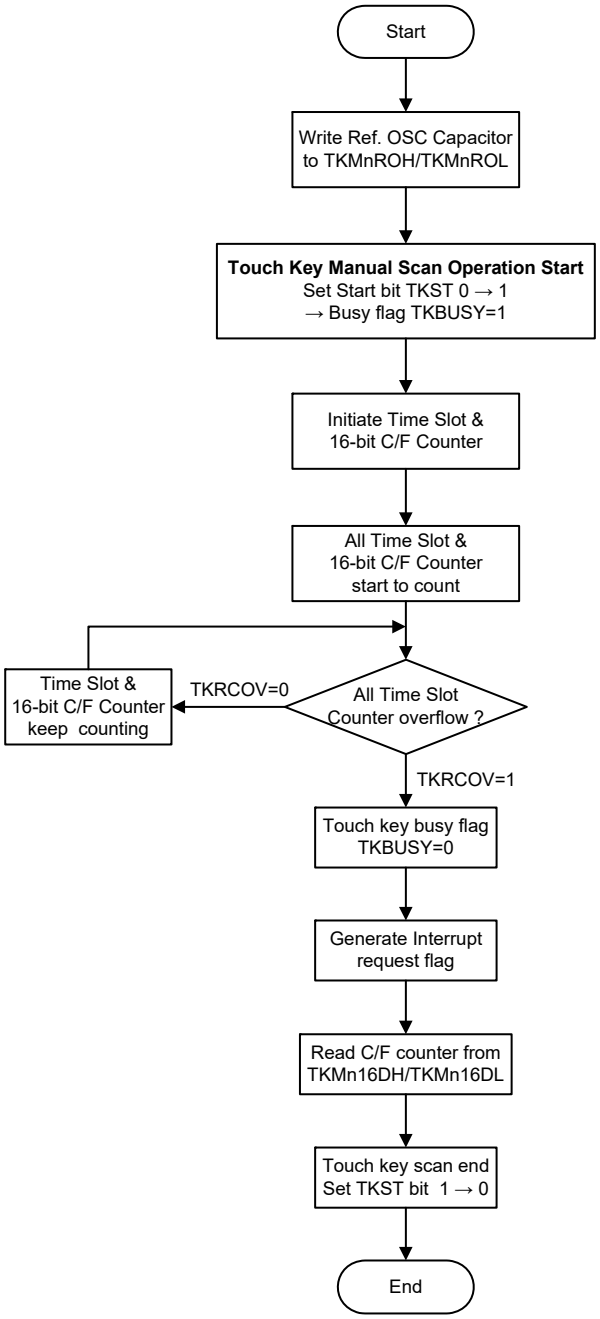
Touch Key Module Data Memory

The device provides two dedicated Data Memory area for the touch key auto scan mode. The addresses 00H~2FH of both Sector 5 and Sector 6 are used by the touch key function, as shown in the following figure. One area is used to store the 16-bit C/F counter values of the touch key module 0~5 and located in Data Memory Sector 5. The other area is used to store the reference oscillator internal capacitor values of the touch key modules and located in Data Memory Sector 6.

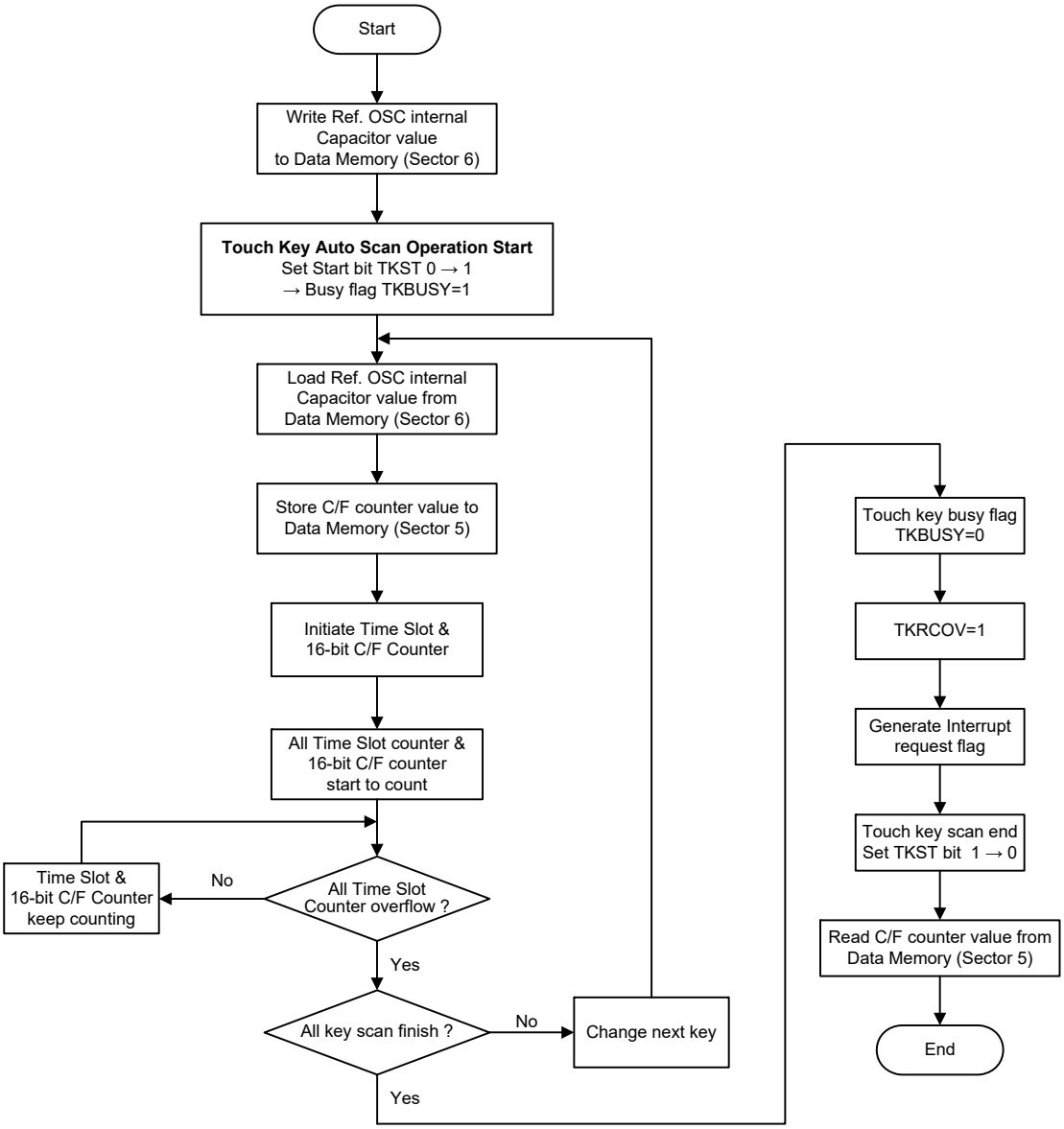


Touch Key Module Data Memory Map

Touch Key Scan Operation Flowchart



Touch Key Manual Scan Mode Flowchart – TKMOD=1, TSCS=0



Touch Key Auto Scan Mode Flowchart – TKMOD=0, TSCS=0

Touch Key Interrupt

The touch key only has a single interrupt, when the time slot counter in all the touch key modules or in the touch key module 0 overflows in manual mode or when all the touch key scan operation is complete in auto scan mode, an actual touch key interrupt will take place. The touch keys mentioned here are the keys which are enabled. The 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter in all modules will be automatically cleared. More details regarding the touch key interrupt is located in the interrupt section of the datasheet.

Progrsmming Considerations

After the relevant registers are set, the touch key detection process is initiated by changing the TKST bit from low to high. This will enable and synchronise all relevant oscillators. The TKRCOV flag which is the time slot counter overflow flag will go high when the counter overflows. When

this happens an interrupt signal will be generated. As the TKRCOV flag will not be automatically cleared, it has to be cleared by the application program.

The TKCFOV flag which is the 16-bit C/F counter overflow flag will go high when any of the Touch Key Module 16-bit C/F counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program. The TK16OV flag which is the 16-bit counter overflow flag will go high when the 16-bit counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program.

When the external touch key size and layout are defined, their related capacitances will then determine the sensor oscillator frequency.

LCD Driver

For large volume applications, which incorporate an LCD in their design, the use of a custom display rather than a more expensive character based display reduces costs significantly. However, the corresponding COM and SEG signals required, which vary in both amplitude and time, to drive such a custom display require many special considerations for proper LCD operation to occur. This device contains an LCD Driver function, which with their internal LCD signal generating circuitry and various options will automatically generate these time and amplitude varying signals to provide a means of direct driving and easy interfacing to a range of custom LCDs.

This device includes a wide range of options to enable LCD displays of various types to be driven. The table shows the range of options available across the device range.

Driver No.	Duty	Bias Level	Bias Type	Waveform Type
43×4	1/4	1/3	R or C	A or B
41×6	1/6	1/3	R or C	A or B
39×8	1/8	1/3	R	A or B
39×8	1/8	1/4	R	A or B

LCD Driver Output Selection

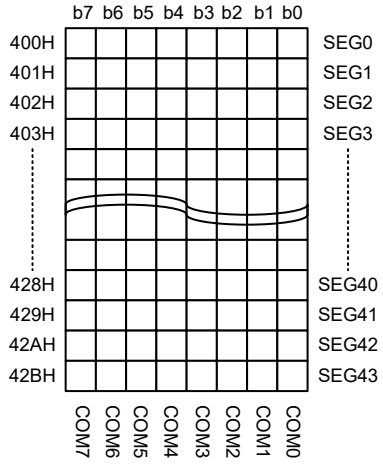
LCD Display Memory

An area of Data Memory is especially reserved for use for the LCD display data. This data area is known as the LCD Memory. Any data written here will be automatically read by the internal display driver circuits, which will in turn automatically generate the necessary LCD driving signals. Therefore any data written into this memory will be immediately reflected into the actual display connected to the microcontroller.

This device provides an area of embedded data memory for the LCD display. This area is located at 00H to 2BH in Sector 4 of the Data Memory. The LCD display memory can be read and written to by indirect addressing mode using MP1L/MP1H and MP2L/MP2H, or by direct addressing mode using the corresponding extended instructions. If using the indirect addressing to access the Display Memory therefore requires first that Sector 4 is selected by writing a value of 04H to MP1H or MP2H. After this, the memory can then be accessed by using indirect addressing through the use of MP1L or MP2L. With Sector 4 selected, then using MP1L/MP2L to read or write to the memory area, from 00H to 2BH, will result in operations to the LCD memory.

When data is written into the display data area, it is automatically read by the LCD driver which then generates the corresponding LCD driving signals. To turn the display on or off, a “1” or a “0” is written to the corresponding bit of the display memory, respectively. The figure illustrates the mapping between the display memory and LCD pattern for the device.

The unimplemented LCD RAM bits cannot be used as general purpose RAM for application. For example, if the LCD duty is selected as 1/4 duty (4-COM), the COM b4~b7 will be read as 0 only.



LCD Memory Map

LCD Clock Source

The LCD clock source is the internal clock signal, f_{LCD}, divided by 8, using an internal divider circuit. The f_{LCD} internal clock is supplied by the LIRC or LXT oscillator, the choice of which is determined by the FSS bit in the SCC register. For proper LCD operation, this arrangement is provided to generate an ideal LCD clock frequency of 4kHz. The on/off state of the LCD clock source f_{LCD} is determined by the LCDEN bit in the LCDC0 register.

f_{LCD} Clock Source	LCD Clock Frequency
LIRC	4kHz
LXT	4kHz

LCD Clock Source

C Type LCD Pump Clock Source

The C type LCD pump clock is generated from the LCD clock source, f_{LCD}, divided by a factor of 2, 4, 8, 16, 32, 64 or 128, using an internal divider circuit. The actual division factor is determined by the LCDPCK2~LCDPCK0 bits in the LCDC2 register.

LCD Registers

Control Registers in the Data Memory, are used to control the various setup features of the LCD Driver. There are three control registers for the LCD function, LCDC0, LCDC1 and LCDC2.

The LCDEN bit in the LCDC0 register provides the overall LCD enable/disable function. Bits RSEL2~RSEL0 in the LCDC0 register select the internal total bias resistors to supply the R type LCD panel with the proper bias current. A choice to best match the LCD panel used in the application can be selected also to minimise bias current. The LCDP1~LCDP0 bits are used to select that the LCD supply power comes from either the external pin or internal power supply for C type bias application. The RCT bit is used to select whether R Type or C Type LCD bias is used. The TYPE bit is used to select whether Type A or Type B LCD waveform signals are used.

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

The LCDC2 register is used for the C type LCD pump clock divider selection as well as LCD duty and bias selection.

Register Name	Bit							
	7	6	5	4	3	2	1	0
LCDC0	TYPE	RCT	LCDP1	LCDP0	RSEL2	RSEL1	RSEL0	LCDEN
LCDC1	QCT2	QCT1	QCT0	—	PLCD3	PLCD2	PLCD1	PLCD0
LCDC2	LCDPCK2	LCDPCK1	LCDPCK0	—	—	DTYC1	DTYC0	BIAS

LCD Register List

- **LCDC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	TYPE	RCT	LCDP1	LCDP0	RSEL2	RSEL1	RSEL0	LCDEN
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 **TYPE**: LCD waveform type selection
0: Type A
1: Type B

Bit 6 **RCT**: LCD bias type selection
0: R type
1: C type

When the RCT bit is cleared to 0, the LCDP[1:0] bits should be fixed at "00" and the power source is from the PLCD pin. As the C1, C2 and V2 pins are pin-shared I/O or other pin-shared functions, when the RCT=1, if the relevant pins are configured to I/O or other pin-shared functions, this will interfere with the C1, C2 and V2 functions.

Bit 5~4 **LCDP1~LCDP0**: C type LCD power source selection
00: From external PLCD/V1/V2 pin input
01: From internal V_C=DPN V_{REF}
10: Forbidden
11: From internal V_A=V_{DD}

The DPN V_{REF} is an internal reference voltage with an approximate level of 1.0V.

Note: The V_{DD} voltage must be greater than or equal to the V_1 or V_A voltage. The "10" setting is forbidden.

Bit 3~1 **RSEL2~RSEL0**: R type total bias resistor selection (for both 1/3 and 1/4 bias)
000: 1170kΩ
001: 225kΩ
010: 60kΩ
011: 2340kΩ
100: Quick charging mode – switching between 60kΩ and 1170kΩ
101: Quick charging mode – switching between 60kΩ and 225kΩ
110~111: Quick charging mode – switching between 60kΩ and 2340kΩ

The device provides the low power quick charging mode for R type LCD display. In quick charging mode the LCD will provide more bias current at the beginning of each COM phase as LCD display refreshes and then provide less bias current to reduce the bias current consumption in the remaining time duration in the same COM phase.

Bit 0 **LCDEN**: LCD enable control
0: Disable
1: Enable

In any system operating mode, the LCD on/off function can be controlled by this bit.

• **LCDC1 Register**

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

Bit 7~5 **QCT2~QCT0**: R type quick charging time selection
000: $1\times t_{LCD}$
001: $2\times t_{LCD}$
010: $3\times t_{LCD}$
011: $4\times t_{LCD}$
100: $5\times t_{LCD}$
101: $6\times t_{LCD}$
110: $7\times t_{LCD}$
111: $8\times t_{LCD}$
Note: $t_{LCD}=1/f_{LCD}$.

Bit 4 Unimplemented, read as "0"

Bit 3~0 **PLCD3~PLCD0**: R type bias supply voltage selection for V_A node
0000: $V_{PLCD}\times 8/16$
0001: $V_{PLCD}\times 9/16$
0010: $V_{PLCD}\times 10/16$
0011: $V_{PLCD}\times 11/16$
0100: $V_{PLCD}\times 12/16$
0101: $V_{PLCD}\times 13/16$
0110: $V_{PLCD}\times 14/16$
0111: $V_{PLCD}\times 15/16$
1xxx: V_{PLCD}
Note: The V_A voltage must be greater than or equal to 2.1V. The V_{DD} voltage must be greater than or equal to the V_A voltage.

• **LCDC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	LCDPCK2	LCDPCK1	LCDPCK0	—	—	DTYC1	DTYC0	BIAS
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7~5 **LCDPCK2~LCDPCK0**: C type LCD Pump clock divider selection
000: 250Hz ($f_{LCD}/128$)
001: 500Hz ($f_{LCD}/64$)
010: 1kHz ($f_{LCD}/32$)
011: 2kHz ($f_{LCD}/16$)
100: 4kHz ($f_{LCD}/8$)
101: 8kHz ($f_{LCD}/4$)
110: 16kHz ($f_{LCD}/2$)
111: 16kHz ($f_{LCD}/2$)
Note: These frequency options are figured out based on the LCD clock source of 32kHz.

Bit 4~3 Unimplemented, read as "0"

Bit 2~1 **DTYC1~DTYC0**: LCD duty selection
00: 1/4 duty – COM0~COM3 are used, for both R and C types
01: 1/6 duty – COM0~COM5 are used, for both R and C types
10: 1/8 duty – COM0~COM7 are used, for R type only
11: Reserved
The unused COM pins can be configured as other pin-shared functions using the corresponding pin-shared selection register.

Bit 0 **BIAS**: LCD bias selection
0: 1/3 bias – for both R and C types
1: 1/4 bias – for R type only

LCD Voltage Source and Biasing

The time and amplitude varying signals generated by the LCD Driver function require the generation of several voltage levels for their operation. The devices can have either R type or C type biasing selected via a software control bit RCT.

R Type Biasing – 1/3 Bias & 1/4 Bias

For R type biasing an external LCD voltage source must be supplied on pin PLCD to generate the internal biasing voltages. This could be the microcontroller power supply or some other voltage source.

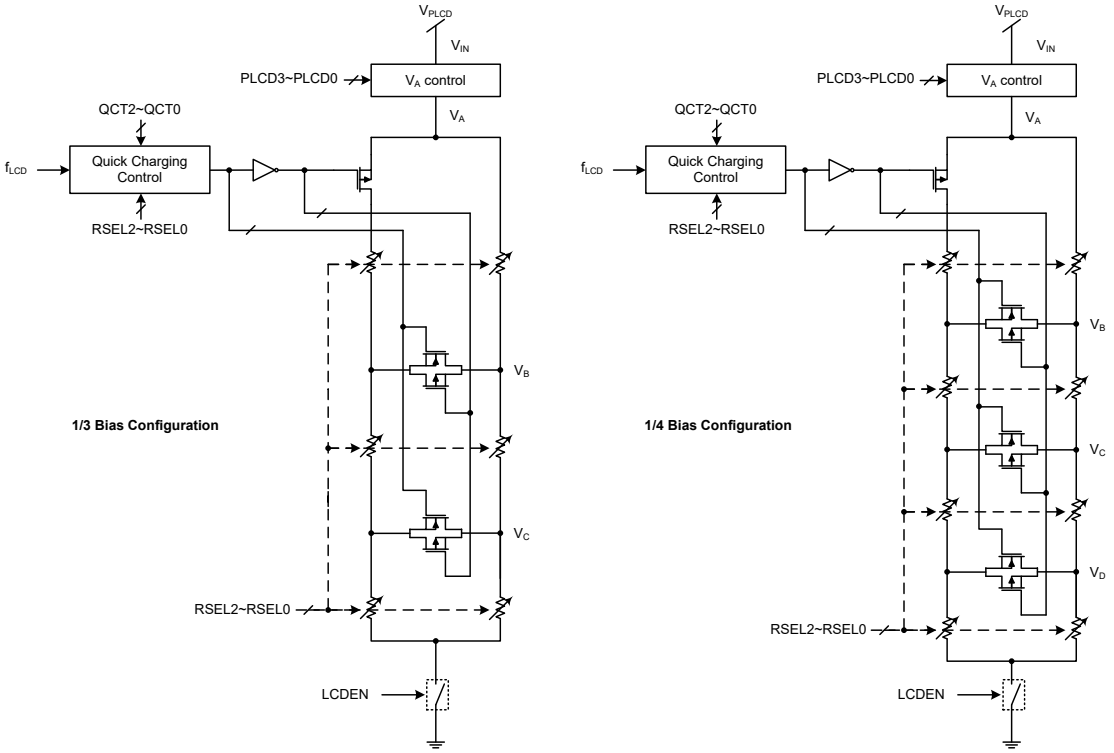
For the R type 1/3 bias scheme, four voltage levels V_{SS}, V_A, V_B and V_C are utilised. For the R type 1/4 bias scheme, five voltage levels V_{SS}, V_A, V_B, V_C and V_D are utilised. The voltage V_A is selected by the PLCD3~PLCD0 bits to be equal to a specific ratio of V_{PLCD} varying from $V_{PLCD}\times8/16$ to V_{PLCD}. Note that the 1/4 bias type is recommended to be used for the 1/8 duty selection.

Different values of internal bias resistors can be selected using the RSEL2~RSEL0 bits in the LCDC0 register. This along with the voltage on pin PLCD will determine the bias current. No external capacitors or resistors are required to be connected if the R type biasing is used. Note that the VMAX pin should be connected to the PLCD or VDD pin which provides the maximum voltage.

Bias Selection	Bias Voltage
1/3 Bias	$V_A=V_{PLCD}\times8/16\sim V_{PLCD}$; $V_B=V_A\times2/3$; $V_C=V_A\times1/3$
1/4 Bias	$V_A=V_{PLCD}\times8/16\sim V_{PLCD}$; $V_B=V_A\times3/4$; $V_C=V_A\times2/4$; $V_D=V_A\times1/4$

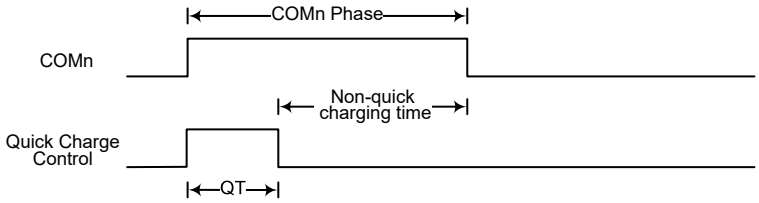
Note: The V_A voltage must be greater than or equal to 2.1V. The V_{DD} voltage must be greater than or equal to the V_A voltage.

R Type Bias Voltage



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

R Type Bias Configurations – 1/3 Bias & 1/4 Bias



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

Quick Charging Mode

C Type Biasing – 1/3 Bias

For C type biasing the LCD voltage source can be supplied on the external pin PLCD, V1 or V2 or derived from the internal power source to generate the required biasing voltages. The C type bias voltage source is selected using the LCDP1 and LCDP0 bits in the LCDC0 register.

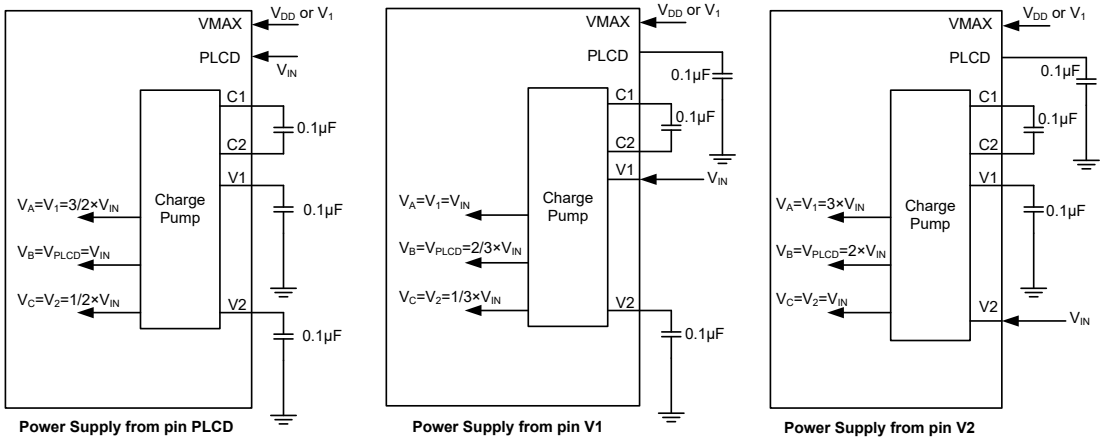
When the LCD voltage source is from the PLCD or V2 pin, the C type biasing scheme uses its internal charge pump circuit, which can generate voltages higher than what is supplied on PLCD or V2. This feature is useful in applications where the microcontroller supply voltage is less than the supply voltage required by the LCD. An additional charge pump capacitor must also be connected between pins C1 and C2 to generate the necessary voltage levels.

For C type 1/3 bias scheme, whether the LCD power is selected from external pin or internal voltage, four voltage levels V_{SS}, V_A, V_B and V_C are utilised. These bias voltages have different levels depending upon different LCD power supply schemes.

LCD Power Supply		V_A Voltage	V_B Voltage	V_C Voltage
External Power Supply	V_{IN} from V1 pin	V_{IN}	$2/3\times V_{IN}$	$1/3\times V_{IN}$
	V_{IN} from PLCD pin	$3/2\times V_{IN}$	V_{IN}	$1/2\times V_{IN}$
	V_{IN} from V2 pin	$3\times V_{IN}$	$2\times V_{IN}$	V_{IN}
Internal Power Supply	V_A (V_A=V_{DD})	V_{DD}	$2/3\times V_{DD}$	$1/3\times V_{DD}$
	V_C (V_C=DPN V_{REF})(2)	$3\times$DPN V_{REF}	$2\times$DPN V_{REF}	DPN V_{REF}

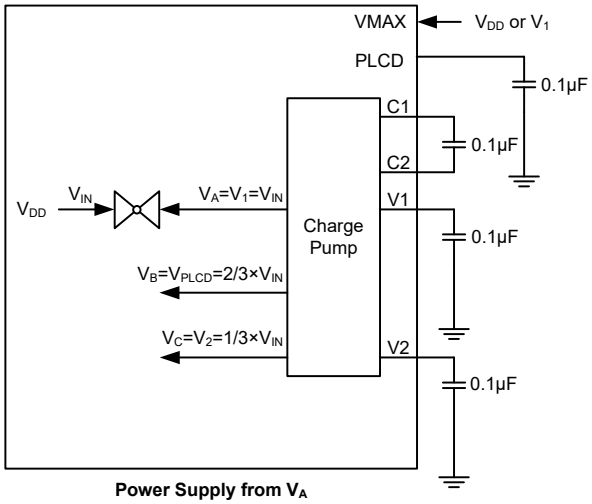
Note: 1. The V_{DD} voltage must be greater than or equal to the V_1 or V_A voltage.
2. The DPN V_{REF} is an internal reference voltage with an approximate level of 1.0V.

C Type Bias Power Supply Scheme

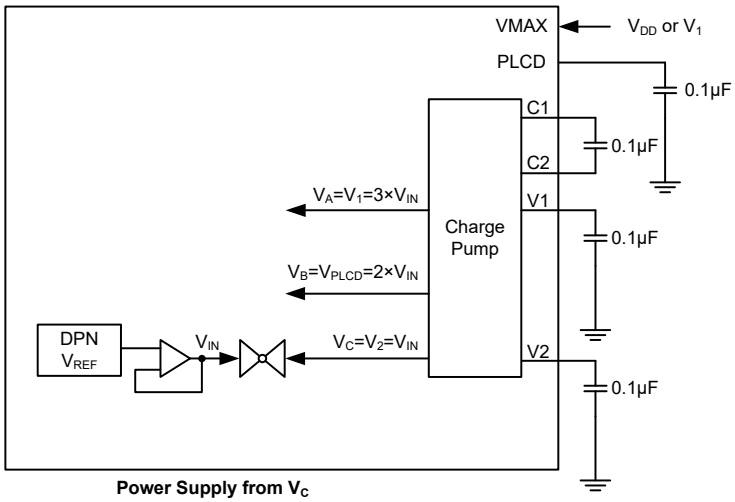


Note: As the V_{DD} voltage must be greater than or equal to the V_1 or V_A voltage, the VMAX pin mus be connected to the maximum voltage V_{DD} to prevent from the pad leakage.

C Type Bias External Power Supply Configuration – 1/3 Bias



Power Supply from V_A



Power Supply from V_C

Note: As the V_{DD} voltage must be greater than or equal to the V_1 or V_A voltage, the VMAX pin mus be connected to the maximum voltage V_{DD} to prevent from the pad leakage.

C Type Bias Internal Power Supply Configuration – 1/3 Bias

LCD Reset Status

The LCD has an internal reset function. Clearing the LCDEN bit to zero will also reset the LCD function. When the LCDEN bit is set to 1 to enable the LCD driver and then an MCU reset occurs, the LCD driver will be reset and the COM and SEG outputs will be in a floating state during the MCU reset duration. The reset operation will take a time of $t_{RSTD}+t_{SST}$. Refer to the System Start Up Time Characteristics for t_{RSTD} and t_{SST} details.

MCU Reset	LCDEN	LCD Reset	COM & SEG Voltage Level
No	1	No	Normal Operation
No	0	Yes	Low
Yes	x	Yes	Floating

"x": Don't care

Note: The Watchdog time-out reset in the IDLE or SLEEP mode is excluded from the MCU Reset conditions.

LCD Reset Status

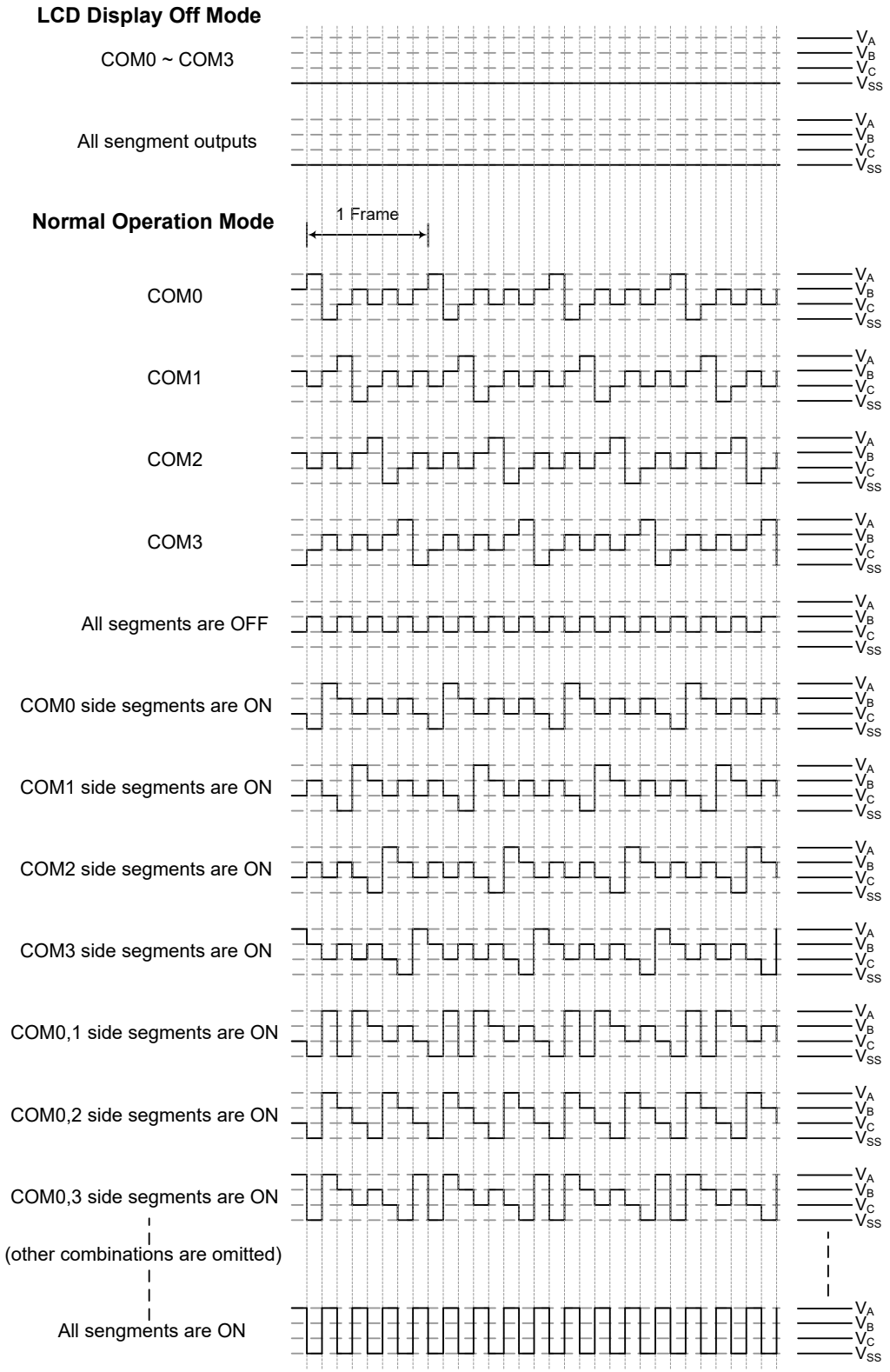
LCD Driver Output

The number of COM and SEG outputs supplied by the LCD driver, as well as its biasing and waveform type selections, are dependent upon how the LCD control bits are programmed.

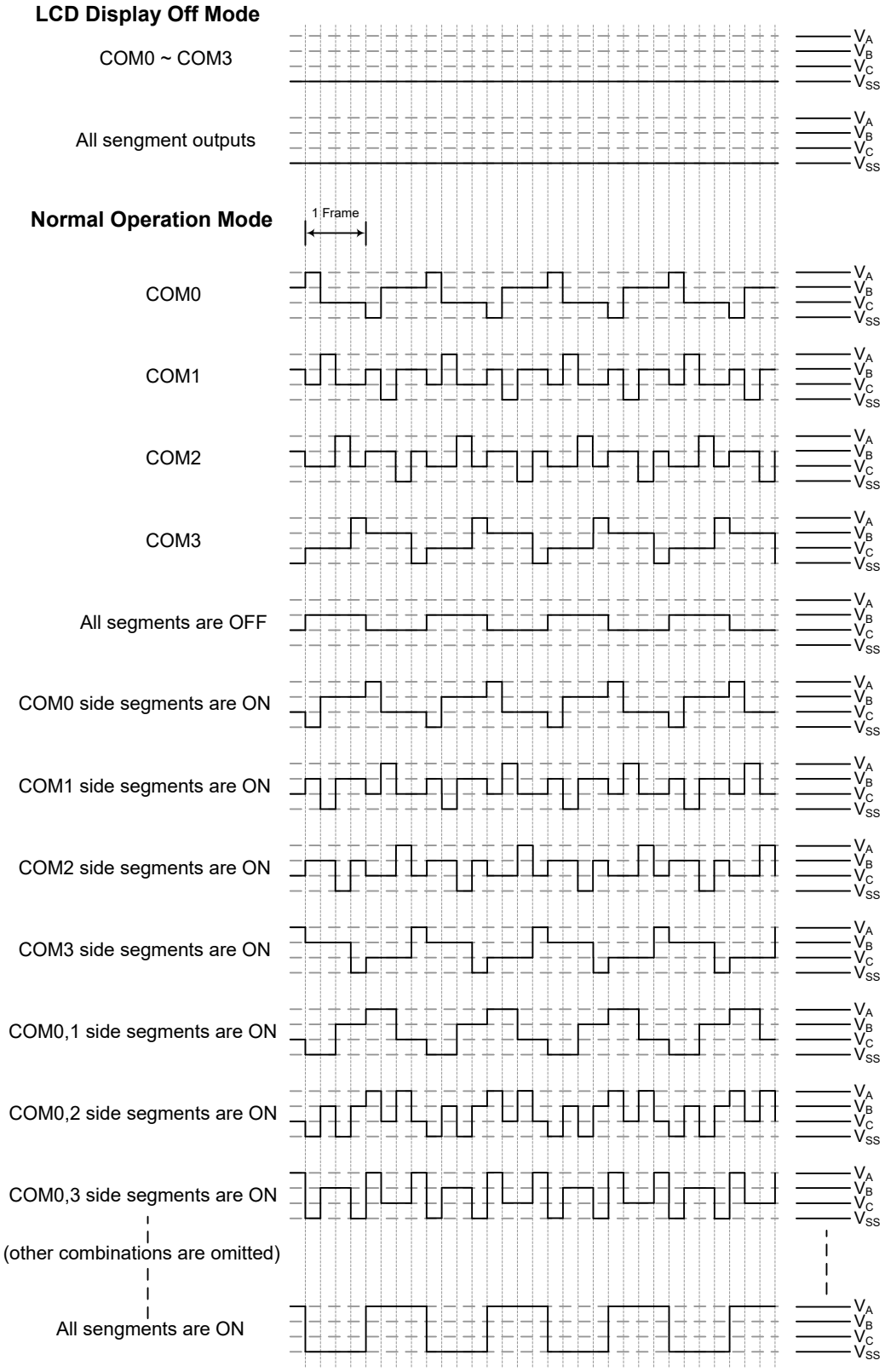
The nature of Liquid Crystal Displays require that only AC voltages can be applied to their pixels as the application of DC voltages to LCD pixels may cause permanent damage. For this reason the relative contrast of an LCD display is controlled by the actual RMS voltage applied to each pixel, which is equal to the RMS value of the voltage on the COM pin minus the voltage applied to the SEG pin. This differential RMS voltage must be greater than the LCD saturation voltage for the pixel to be on and less than the threshold voltage for the pixel to be off.

The requirement to limit the DC voltage to zero and to control as many pixels as possible with a minimum number of connections requires that both a time and amplitude signal is generated and applied to the application LCD. These time and amplitude varying signals are automatically generated by the LCD driver circuits in the microcontroller. What is known as the duty determines the number of common lines used, which are also known as backplanes or COMs. For example, the duty is 1/4 and equates to a COM number of 4, therefore defines the number of time divisions within each LCD signal frame. Two types of signal generation are also provided, known as Type A and Type B, the required type is selected via the TYPE bit in the LCDC0 register. Type B offers lower frequency signals, however lower frequencies may introduce flickering and influence display clarity.

R & C Type, 4-COM, 1/3 Bias

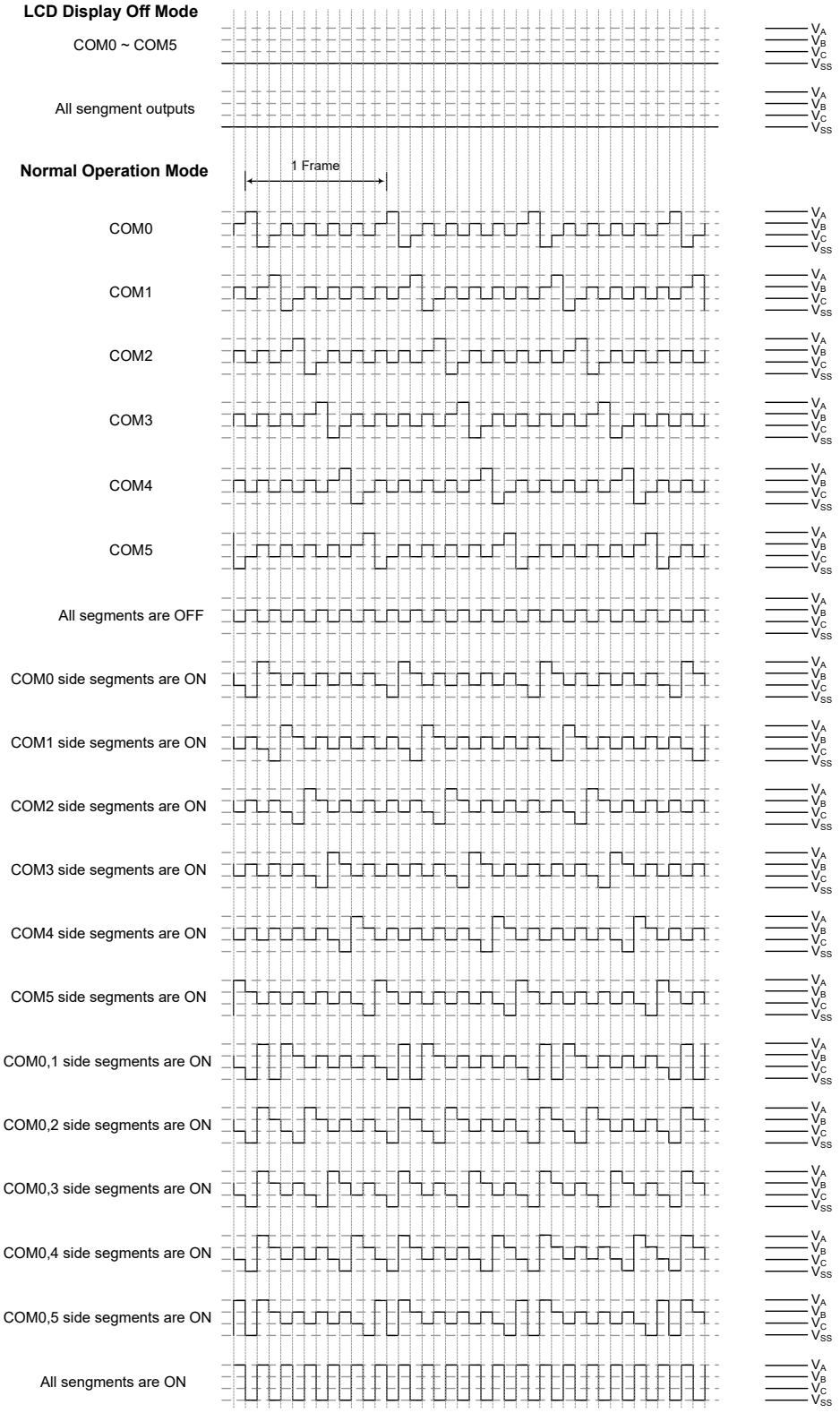


LCD Driver Output – Type A, 1/4 Duty, 1/3 Bias

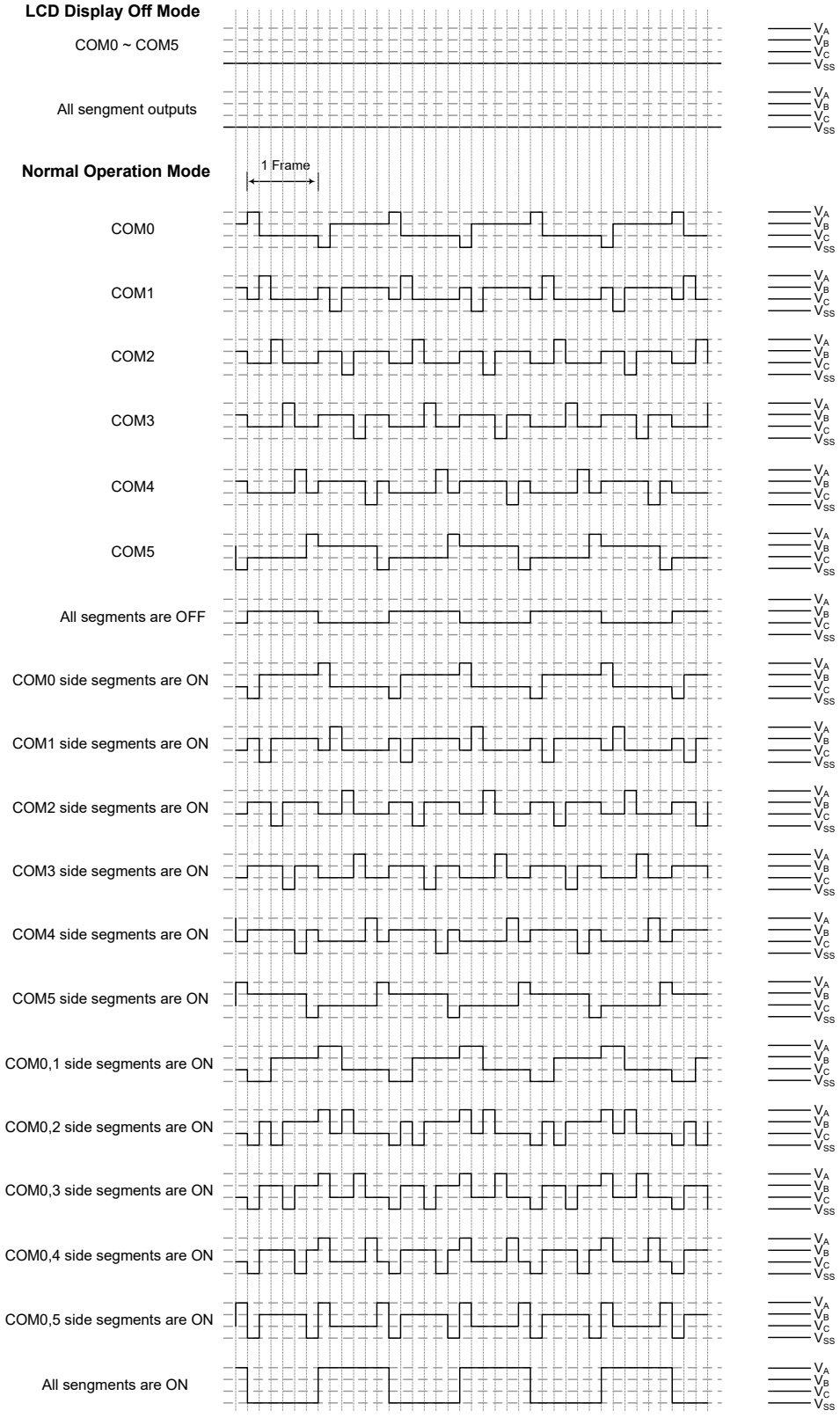


LCD Driver Output – Type B, 1/4 Duty, 1/3 Bias

R & CType, 6-COM, 1/3 Bias

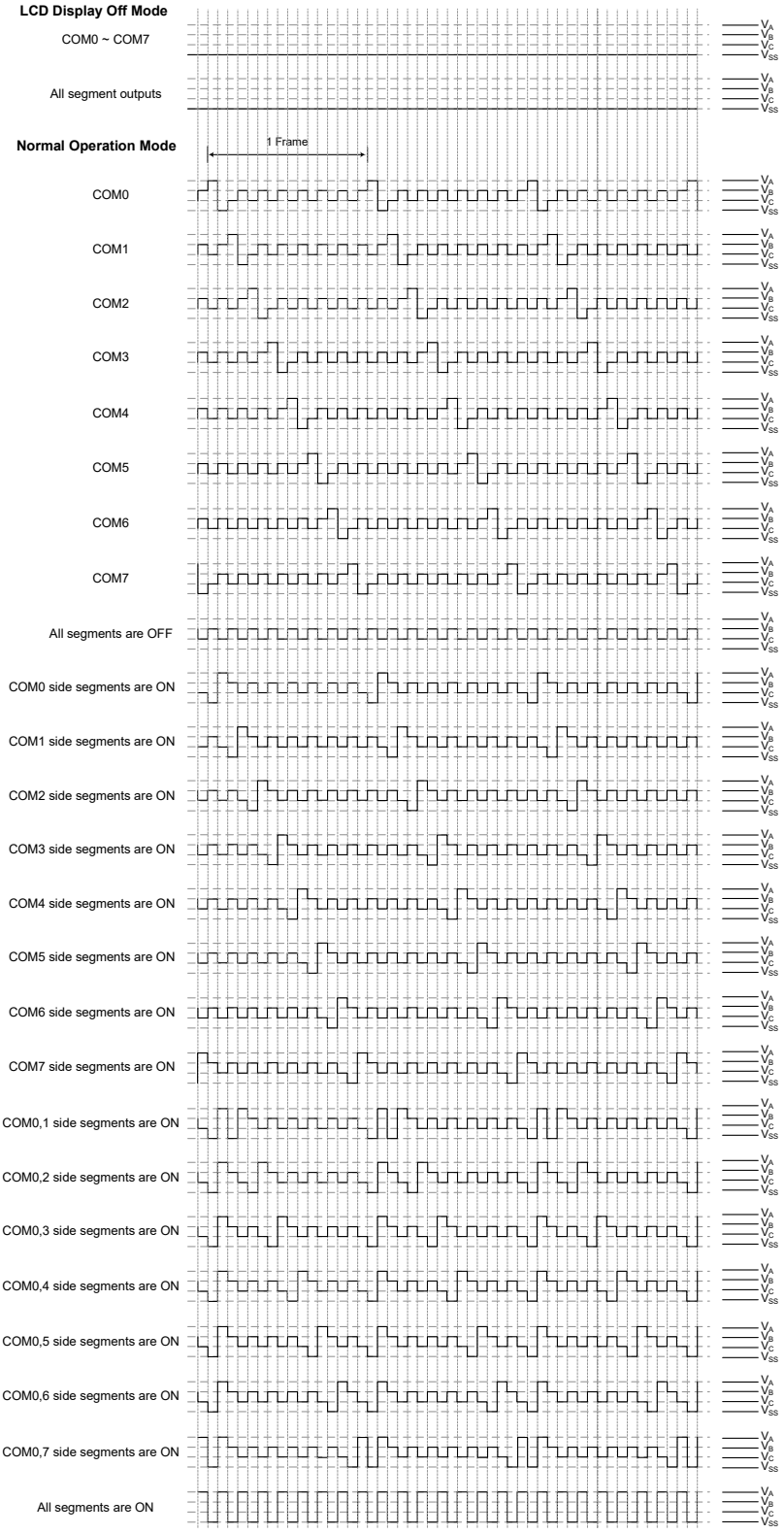


LCD Driver Output – Type A, 1/6 Duty, 1/3 Bias

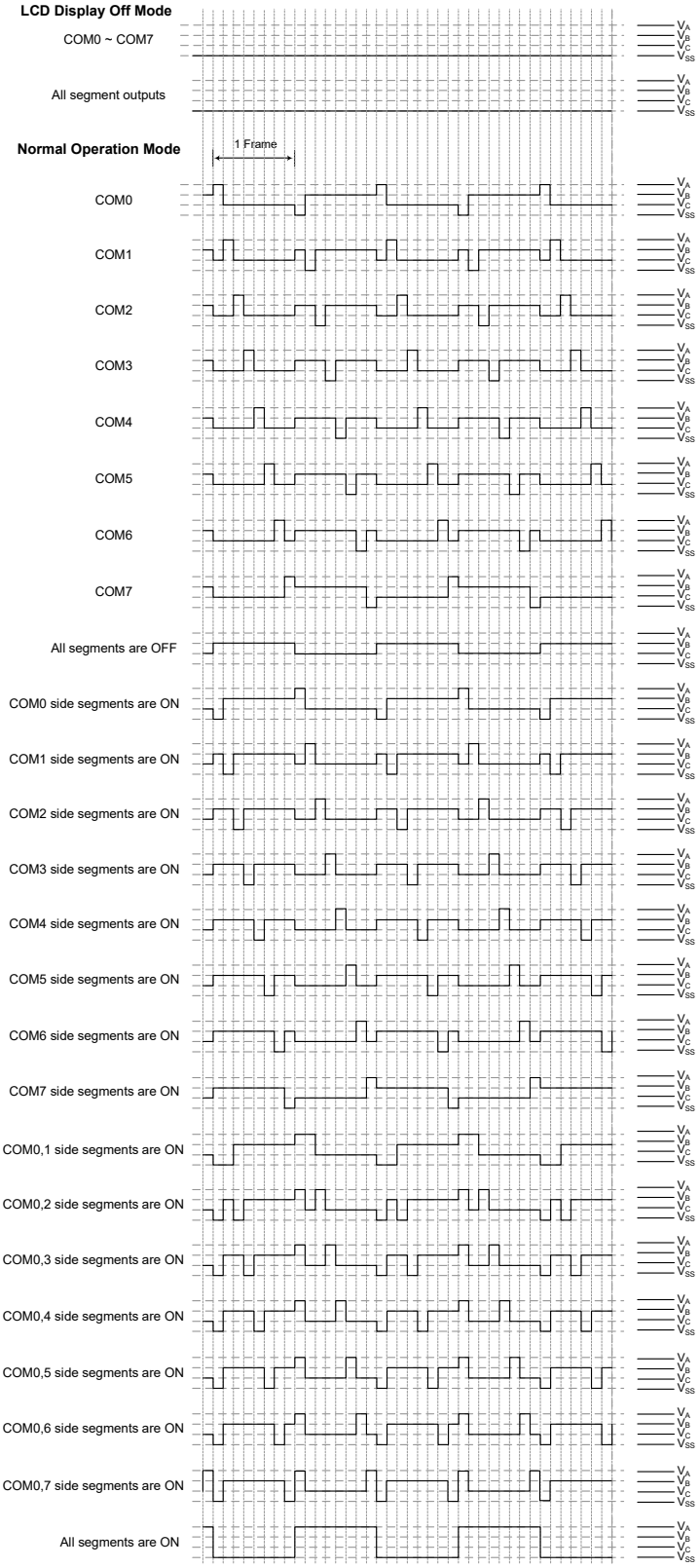


LCD Driver Output – Type B, 1/6 Duty, 1/3 Bias

R Type, 8-COM, 1/3 Bias

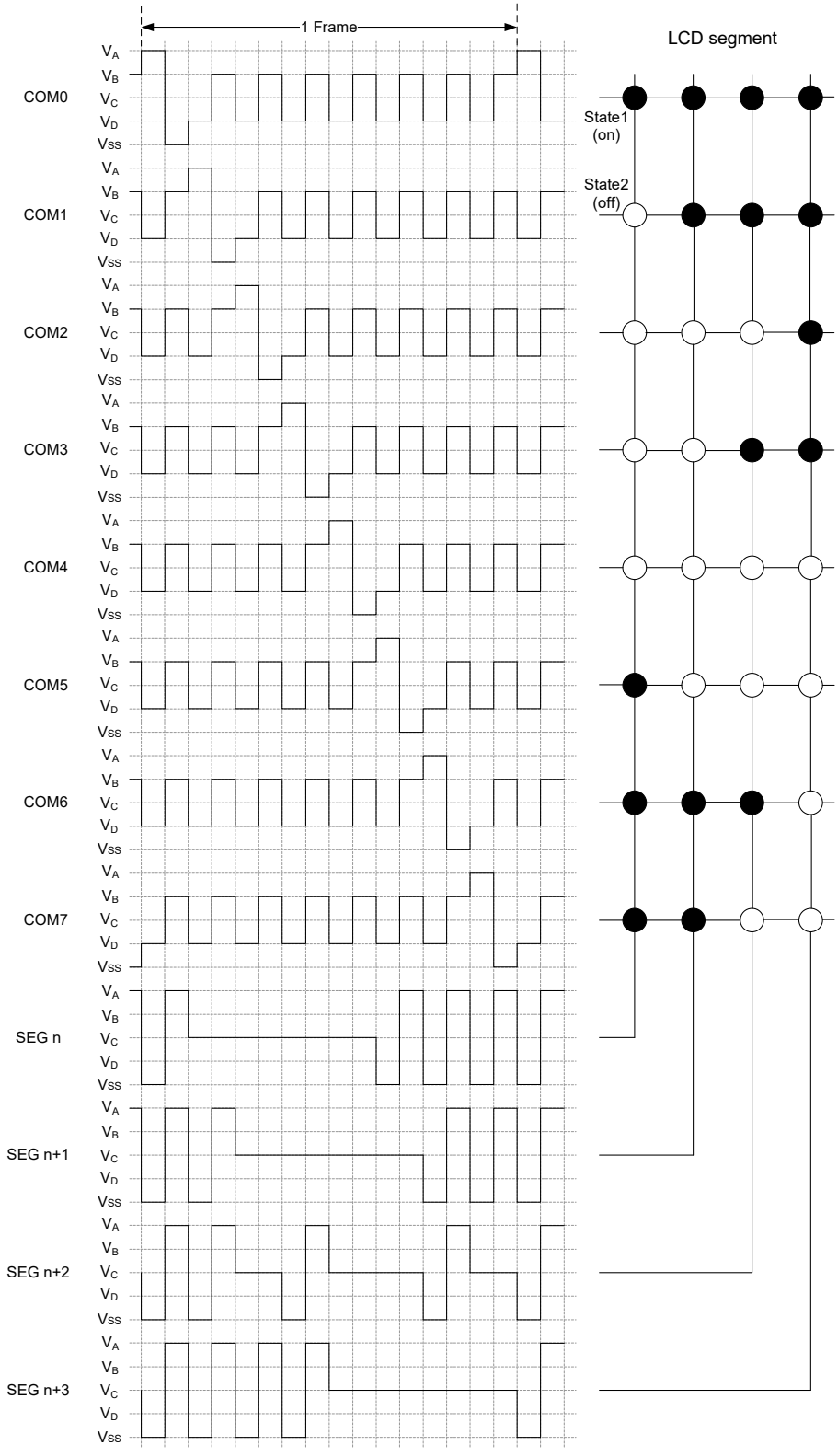


LCD Driver Output – Type A, 1/8 Duty, 1/3 Bias

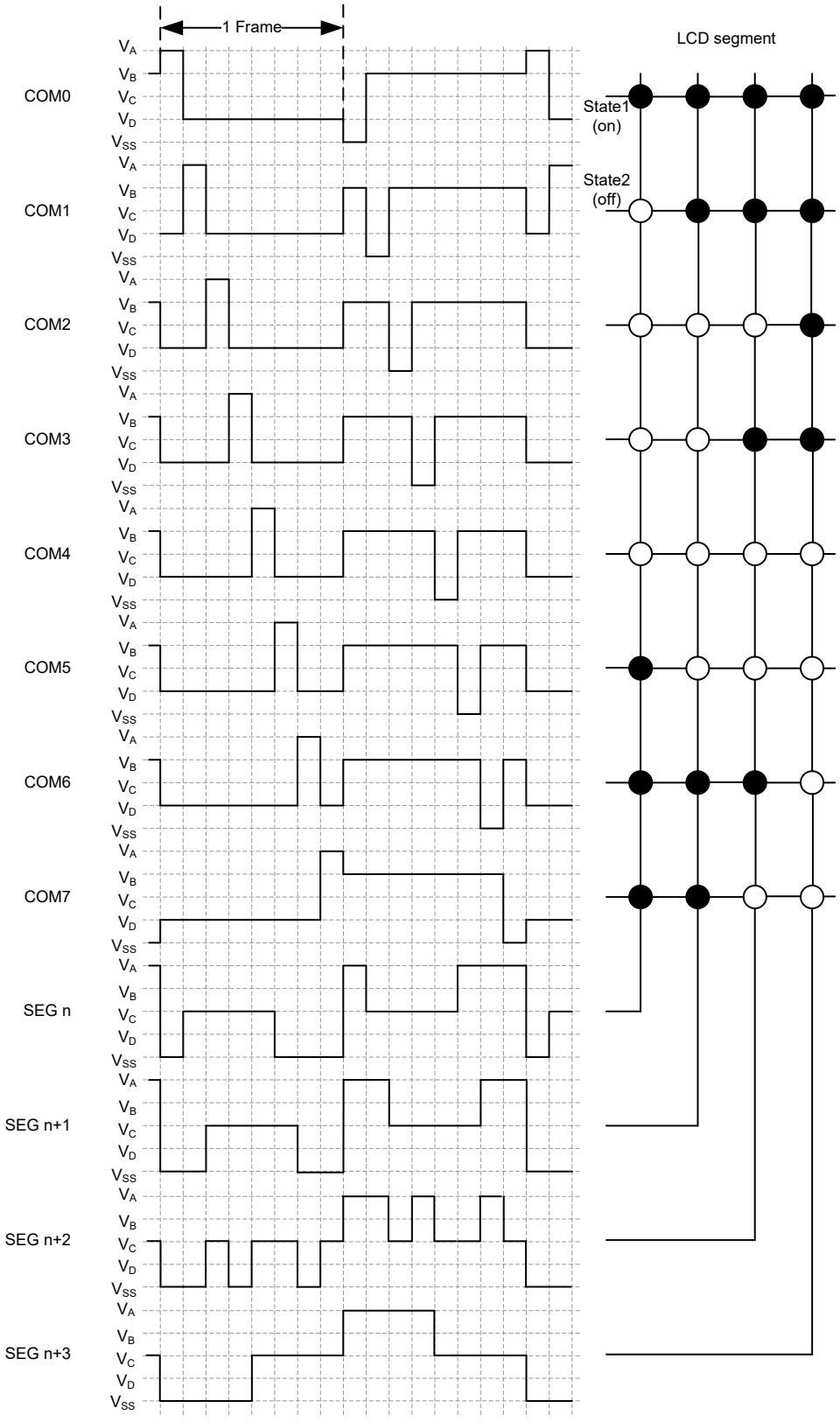


LCD Driver Output – Type B, 1/8 Duty, 1/3 Bias

R Type, 8-COM, 1/4 Bias



LCD Driver Output – Type A, 1/8 Duty, 1/4 Bias



LCD Driver Output – Type B, 1/8 Duty, 1/4 Bias

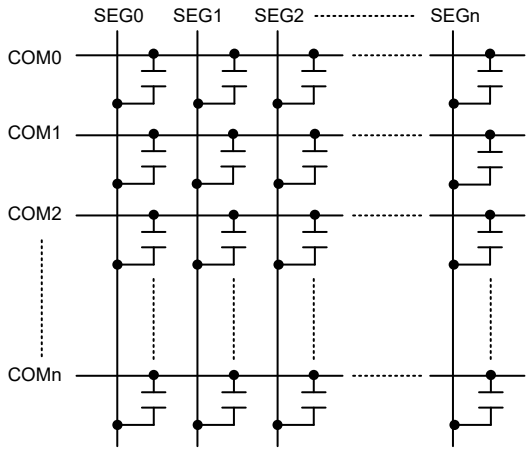
Programming Considerations

Certain precautions must be taken when programming the LCD. One of these is to ensure that the LCD Memory is properly initialised after the microcontroller is powered on. Like the General Purpose Data Memory, the contents of the LCD Memory are in an unknown condition after power-on. As the contents of the LCD Memory will be mapped into the actual display, it is important to initialise this memory area into a known condition soon after applying power to obtain a proper display pattern.

Consideration must also be given to the capacitive load of the actual LCD used in the application. As the load presented to the microcontroller by LCD pixels can be generally modeled as mainly capacitive in nature, it is important that this is not excessive, a point that is particularly true in the case of the COM lines which may be connected to many LCD pixels. The accompanying diagram depicts the equivalent circuit of the LCD.

It must be noted that the on/off state of the LCD clock source f_{LCD} is controlled by the LCDEN bit in the LCDC0 regsiter, which also permits the display to be powered off to reduce power consumption. If this bit is zero, the driving signals to the display will cease, producing a blank display pattern but reducing any power consumption associated with the LCD.

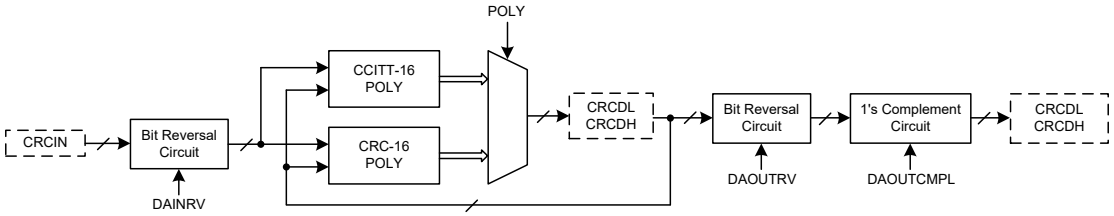
After Power-on, note that as the LCDEN bit is cleared to zero, the display function will be disabled.



LCD Panel Equivalent Circuit

Cyclic Redundancy Check – CRC

The Cyclic Redundancy Check, CRC, calculation unit is an error detection technique test algorithm and is used 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 which is used as a checksum when being sent or stored. Therefore, the received or restored data stream is calculated by the same generator polynomial. If the new CRC code result does not match the one calculated earlier, that means data stream contains a data error.



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 determine which CRC generating polynomial is used and whether to execute the bit reversal operation and 1's complement operation.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CRCCR	—	—	—	—	DAINRV	DAOUTRV	DAOUTCMPL	POLY
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D15	D14	D13	D12	D11	D10	D9	D8

CRC Register List

• CRCCR Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	DAINRV	DAOUTRV	DAOUTCMPL	POLY
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **DAINRV**: Bit reversal operation control on input data
0: Disable
1: Enable

Bit 2 **DAOUTRV**: Bit reversal operation control on checksum output
0: Disable
1: Enable

Bit 1 **DAOUTCMPL**: 1's complement operation control on checksum output
0: Disable – final XOR value=0000H
1: Enable – final XOR value=FFFFH

Bit 0 **POLY**: 16-bit CRC generating polynomial selection
0: Polynomial: 1021H, $X^{16}+X^{12}+X^5+1$
1: Polynomial: 8005H, $X^{16}+X^{15}+X^2+1$

Register/Bit		POLY	CRCDx	DAOUTCMPL	DAINRV	DAOUTRV
CRC Algorithm	Polynomial Formula	Polynomial	Initial Value	Final XOR Value	Input Bit Reverse	Output Bit Reverse
CRC-16-IBM	$X^{16}+X^{15}+X^2+1$	8005H	0000H	0000H	True	True
CRC-16-MAXIM	$X^{16}+X^{15}+X^2+1$	8005H	0000H	FFFFH	True	True
CRC-16-USB	$X^{16}+X^{15}+X^2+1$	8005H	FFFFH	FFFFH	True	True
CRC-16-MODBUS	$X^{16}+X^{15}+X^2+1$	8005H	FFFFH	0000H	True	True
CRC-16-CCITT	$X^{16}+X^{12}+X^5+1$	1021H	0000H	0000H	True	True
CRC-16-CCITT-FALSE	$X^{16}+X^{12}+X^5+1$	1021H	FFFFH	0000H	False	False
CRC-16-X25	$X^{16}+X^{12}+X^5+1$	1021H	FFFFH	FFFFH	True	True
CRC-16-XMODEM	$X^{16}+X^{12}+X^5+1$	1021H	0000H	0000H	False	False

- **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

CRC Operation

The CRC generator provides the 16-bit CRC calculation based on the 8005H ($X^{16}+X^{15}+X^2+1$) and 1021H ($X^{16}+X^{12}+X^5+1$) 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.

- Polynomial: 1021H, $X^{16}+X^{12}+X^5+1$
- Polynomial: 8005H, $X^{16}+X^{15}+X^2+1$

CRC Operation Steps

- Step 1

  Set the CRC initial value by setting the CRCDH and CRCDL register pair to 0000H or FFFFH. (CRCDH=CRCDL=00H or FFH).

- Step 2

  Set the DAINRV bit to determine whether to execute a bit reversal operation on the 8-bit input data.

- Step 3

  Set the DAOUTRV bit to determine whether to execute a bit reversal operation on the 16-bit output data.

- Step 4

  Set the DAOUTCMPL bit to determine whether to execute a 1's complement operation on the 16-bit output data.

- Step 5

  Set the POLY bit to determine the polynomial value to be either 1021H or 8005H.

- Step 6

  After the input data is written into the CRCIN register, wait for one instruction cycle to complete the CRC computation. If there is more than one data byte, repeat step 6 until all the data has been written. The final CRC value can be obtained by reading the CRCDH and CRCDL registers.

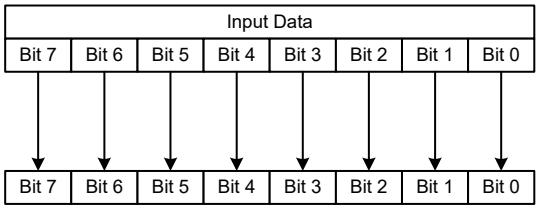
CRC Computation

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 generator calculates the CRC data register value on a byte-by-byte basis. It will take one MCU instruction cycle to calculate the CRC checksum.

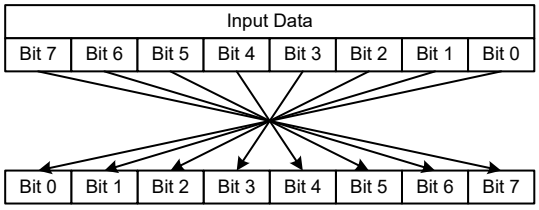
Input Data and Checksum Output Bit Reversal Operation

Input Data Bit Reversal Operation

- When the DAINRV bit is "0", no bit reversal operation on the 8-bit input data, as shown in the following diagram:

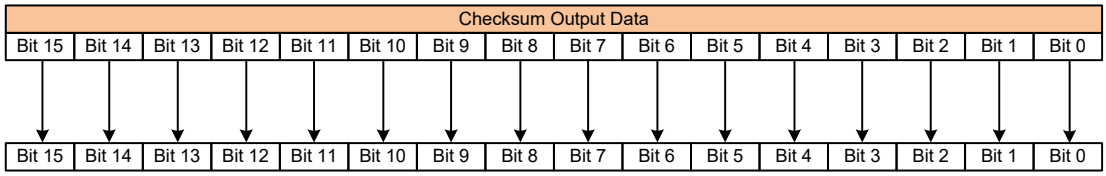


- When the DAINRV bit is "1", execute a bit reversal operation on the 8-bit input data, as shown in the following diagram:

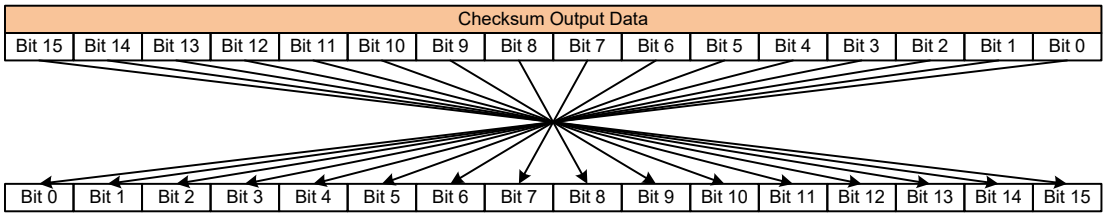


Checksum Output Bit Reversal Operation

- When the DAOUTRV bit is "0", no bit reversal operation on the 16-bit checksum output data, as shown in the following diagram:



- When the DAOUTRV bit is "1", execute a bit reversal operation on the 16-bit checksum output data, as shown in the following diagram:



1's Complement on Checksum Output

- When the DAOUTCMPL bit is low, execute an "Exclusive OR" operation with the 16-bit checksum output data and 0000H.
- When the DAOUTCMPL bit is high, execute an "Exclusive OR" operation with the 16-bit checksum output data and FFFFH.

CRC Calculation Steps

- Step 1

  Set the CRC 16-bit initial value to 0000H or FFFFH.

- Step 2

  Determine whether to execute a bit reversal operation on the 8-bit input data.

  If yes, execute a bit reversal operation on the 8-bit input data. If no, no bit reversal operation is executed on the 8-bit input data. The result of this step is called CRCDATA.

- Step 3

  Execute an "Exclusive OR" operation with the CRCDATA and the high byte of the 16-bit initial value. The result is called the temporary CRCSUM.

- Step 4

  Shift the temporary CRCSUM value left by one bit and move a "0" into the LSB.

- Step 5

  Check the shifted temporary CRCSUM value after step 4.

  If the MSB is 0, then this shifted temporary CRCSUM will be considered as a new temporary CRCSUM. If the MSB is 1, execute an "Exclusive OR" operation with the shifted temporary CRCSUM in step 4 and a data 8005H or 1021H. Then the operation result will be regarded as the new temporary CRCSUM.

- Step 6

  Repeat step 4 ~ step 5 until all bits of the input data byte are completely calculated.

- Step 7

  Repeat step 2 ~ step 6 until all of the input data bytes are completely calculated.

- Step 8

  Determine whether to execute a bit reversal operation on the 16-bit temporary CRCSUM in step 6 or step 7.

  If yes, execute a bit reversal operation on the 16-bit temporary CRCSUM. If no, no bit reversal operation is executed on the 16-bit temporary CRCSUM. The result of this step is called 16-bit CRCSUM1.

- Step 9

  Determine whether to execute a 1's complement operation on the 16-bit CRCSUM1.

  If yes, execute an "Exclusive OR" operation with the 16-bit CRCSUM1 and FFFFH. If no, execute an "Exclusive OR" operation with the 16-bit CRCSUM1 and 0000H. The result of this step is called 16-bit CRCSUMF.

- Step 10

  The 16-bit CRCSUMF value is the final CRC calculation result, which is stored into the CRCDH and CRCDL registers.

CRC Calculation Examples

- **Example 1: write 1 byte input data into the CRCIN register to calculate the CRC value and the corresponding settings are shown below.**

1. The initial value of the CRC checksum register pair, CRCDH and CRCDL, is cleared to 0000H.
2. DAINRV=0 – A bit reversal operation on the input data is not executed.
3. DAOUTRV=0 – A bit reversal operation on the output data is not executed.
4. DAOUTCMPL=0 – A 1's complement operation on the output data is not executed.

The input and output results are listed in the following table, where the corresponding CRC checksum are individually calculated.

CRC Data Input / CRC Polynomial	00H	01H	02H	03H	04H	05H	06H	07H
1021H ($X^{16}+X^{12}+X^5+1$)	0000H	1021H	2042H	3063H	4084H	50A5H	60C6H	70E7H
8005H ($X^{16}+X^{15}+X^2+1$)	0000H	8005H	800FH	000AH	801BH	001EH	0014H	8011H

- **Example 2: write 4 bytes input data into the CRCIN register sequentially to calculate the CRC value and the corresponding settings are shown below.**

1. The initial value of the CRC checksum register pair, CRCDH and CRCDL, is cleared to 0000H.
2. DAINRV=0 – A bit reversal operation on the input data is not executed.
3. DAOUTRV=0 – A bit reversal operation on the output data is not executed.
4. DAOUTCMPL=0 – A 1's complement operation on the output data is not executed.

The input and output results are sequentially listed in the following table.

CRC Data Input / CRC Polynomial	CRCIN = 78H→56H→34H→12H
1021H ($X^{16}+X^{12}+X^5+1$)	(CRCDH, CRCDL)=FF9FH→BBC3H→A367H→D0FAH
8005H ($X^{16}+X^{15}+X^2+1$)	(CRCDH, CRCDL)=0110H→91F1H→F2DEH→5C43H

- **Example 3: write 4 bytes input data into the CRCIN register sequentially to calculate the CRC value and the corresponding settings are shown below.**

  1. The initial value of the CRC checksum register pair, CRCDH and CRCDL, is set to FFFFH.
  2. DAINRV=1 – A bit reversal operation on the input data is executed.
  3. DAOUTRV=1 – A bit reversal operation on the output data is executed.
  4. DAOUTCMPL=0 – A 1's complement operation on the output data is not executed.

  The input and output results are sequentially listed in the following table.

CRC Data Input / CRC Polynomial	CRCIN=78H→56H→34H→12H
8005H ($X^{16}+X^{15}+X^2+1$)	(CRCDH, CRCDL) = 62BFH→8EA3H→AECFH→596EH

- **Example 4: write 4 bytes input data into the CRCIN register sequentially to calculate the CRC value and the corresponding settings are shown below.**

  1. The initial value of the CRC checksum register pair, CRCDH and CRCDL, is cleared to 0000H.
  2. DAINRV=1 – A bit reversal operation on the input data is executed.
  3. DAOUTRV=1 – A bit reversal operation on the output data is executed.
  4. DAOUTCMPL=0 – A 1's complement operation on the output data is not executed.

  The input and output results are sequentially listed in the following table.

CRC Data Input / CRC Polynomial	CRCIN=78H→56H→34H→12H
1021H ($X^{16}+X^{12}+X^5+1$)	(CRCDH, CRCDL)=FFCFH→09B7H→B69AH→08F6H

- **Example 5: write 4 bytes input data into the CRCIN register sequentially so as to calculate the CRC value and the corresponding settings are shown below.**

  1. The initial value of the CRC checksum register pair, CRCDH and CRCDL, is cleared to 0000H.
  2. DAINRV=1 – A bit reversal operation on the input data is executed.
  3. DAOUTRV=1 – A bit reversal operation on the output data is executed.
  4. DAOUTCMPL=1 – A 1's complement operation on the output data is executed.

  The input and output results are sequentially listed in the following table.

CRC Data Input / CRC Polynomial	CRCIN=78H→56H→34H→12H
8005H ($X^{16}+X^{15}+X^2+1$)	(CRCDH, CRCDL)=DDFFH→C15DH→9141H→8291H

Low Voltage Detector – LVD

Each 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**: Select Minimum low voltage width to interrupt (t_{LVD})
00: (1~2)×t_{SUB}
01: (3~4)×t_{SUB}
10: (7~8)×t_{SUB}
11: (1~2)×t_{SUB}

Bit 5 **LVDO**: LVD output flag
0: No low voltage detected
1: Low voltage detected

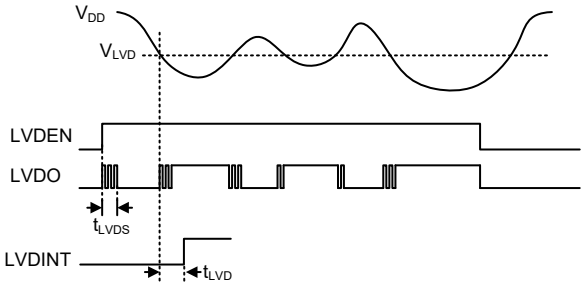
Bit 4 **LVDEN**: Low voltage detector enable control
0: Disable
1: Enable

Bit 3 **VBGEN**: Bandgap reference enable control
0: Disable
1: Enable
Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VBGEN bit is set high.

Bit 2~0 **VLVD2~VLVD0**: LVD voltage selection
000: 1.8V
001: 2.0V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a voltage level pre-specified by the LVDC register. This has a range of 1.8V~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 enters the SLEEP mode, the low voltage detector will be automatically disabled even if the LVDEN bit is set 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} may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



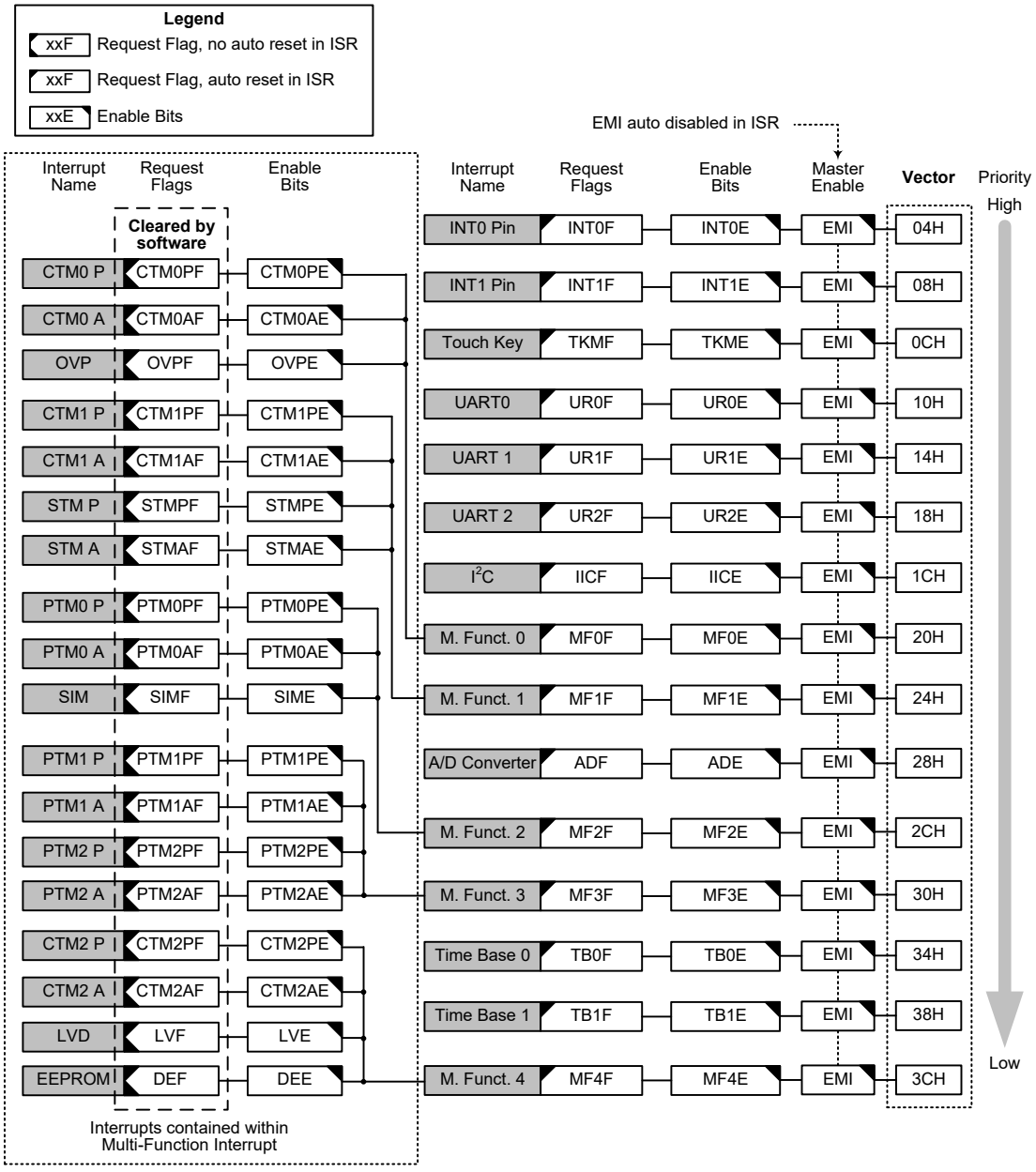
LVD Operation

The Low Voltage Detector also has its own interrupt, 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 enters the IDLE Mode.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Bases, EEPROM, SIM, I^2C, UART, OVP, LVD, A/D converter, and Touch Key etc.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector.



Interrupt 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, as shown in the accompanying table. The registers fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI4 registers which setup the Multi-function interrupts. Finally there is a register, INTEG, 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
Touch Key Module	TKME	TKMF	—
UART Interfaces	URnE	URnF	n=0~2
I^2C Interface	IICE	IICF	—
A/D Converter	ADE	ADF	—
Time Base	TBnE	TBnF	n=0~1
Multi-function	MFnE	MFnF	n=0~4
EEPROM erase or write operation	DEE	DEF	—
LVD	LVE	LVF	—
SIM	SIME	SIMF	—
OVP	OVPE	OVPF	—
CTM	CTMnPE	CTMnPF	n=0~2
	CTMnAE	CTMnAF	n=0~2
STM	STMPE	STMPF	—
	STMAE	STMAF	—
PTM	PTMnPE	PTMnPF	n=0~2
	PTMnAE	PTMnAF	n=0~2

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	TKMF	INT1F	INT0F	TKME	INT1E	INT0E	EMI
INTC1	IICF	UR2F	UR1F	UR0F	IICE	UR2E	UR1E	UR0E
INTC2	MF2F	ADF	MF1F	MF0F	MF2E	ADE	MF1E	MF0E
INTC3	MF4F	TB1F	TB0F	MF3F	MF4E	TB1E	TB0E	MF3E
MFI0	—	OVPF	CTM0AF	CTM0PF	—	OVPE	CTM0AE	CTM0PE
MFI1	STMAF	STMPF	CTM1AF	CTM1PF	STMAE	STMPE	CTM1AE	CTM1PE
MFI2	—	SIMF	PTM0AF	PTM0PF	—	SIME	PTM0AE	PTM0PE
MFI3	PTM2AF	PTM2PF	PTM1AF	PTM1PF	PTM2AE	PTM2PE	PTM1AE	PTM1PE
MFI4	CTM2AF	CTM2PF	DEF	LVF	CTM2AE	CTM2PE	DEE	LVE

Interrupt Register List

- **INTEG Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt edge control for INT0 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TKMF	INT1F	INT0F	TKME	INT1E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **TKMF**: Touch Key Module interrupt request flag
0: No request
1: Interrupt request

Bit 5 **INT1F**: INT1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INT0F**: INT0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TKME**: Touch Key Module interrupt control
0: Disable
1: Enable

Bit 2 **INT1E**: INT1 interrupt control
0: Disable
1: Enable

Bit 1 **INT0E**: INT0 interrupt control
0: Disable
1: Enable

Bit 0 **EMI**: Global interrupt control
0: Disable
1: Enable

• **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	IICF	UR2F	UR1F	UR0F	IICE	UR2E	UR1E	UR0E
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 **IICF**: I^2C interrupt request flag
0: No request
1: Interrupt request

Bit 6 **UR2F**: UART2 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **UR1F**: UART1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **UR0F**: UART0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **IICE**: I²C interrupt control
0: Disable
1: Enable

Bit 2 **UR2E**: UART2 interrupt control
0: Disable
1: Enable

Bit 1 **UR1E**: UART1 interrupt control
0: Disable
1: Enable

Bit 0 **UR0E**: UART0 interrupt control
0: Disable
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	MF2F	ADF	MF1F	MF0F	MF2E	ADE	MF1E	MF0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **MF2F**: Multi-function interrupt 2 request flag
0: No request
1: Interrupt request

Bit 6 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 5 **MF1F**: Multi-function interrupt 1 request flag
0: No request
1: Interrupt request

Bit 4 **MF0F**: Multi-function interrupt 0 request flag
0: No request
1: Interrupt request

Bit 3 **MF2E**: Multi-function interrupt 2 control
0: Disable
1: Enable

Bit 2 **ADE**: A/D Converter interrupt control
0: Disable
1: Enable

Bit 1 **MF1E**: Multi-function interrupt 1 control
0: Disable
1: Enable

Bit 0 **MF0E**: Multi-function interrupt 0 control
0: Disable
1: Enable

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	MF4F	TB1F	TB0F	MF3F	MF4E	TB1E	TB0E	MF3E
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 **MF4F**: Multi-function interrupt 4 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 **MF3F**: Multi-function interrupt 3 request flag
0: No request
1: Interrupt request

Bit 3 **MF4E**: Multi-function interrupt 4 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 **MF3E**: Multi-function interrupt 3 control
0: Disable
1: Enable

• **MFI0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	OVPF	CTM0AF	CTM0PF	—	OVPE	CTM0AE	CTM0PE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Umimplemented, read as "0"

Bit 6 **OVPF**: Over Voltage Protection 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 **CTM0AF**: CTM0 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 **CTM0PF**: CTM0 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 Umimplemented, read as "0"

Bit 2 **OVPE**: Over Voltage Protection interrupt control
0: Disable
1: Enable

Bit 1 **CTM0AE**: CTM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **CTM0PE**: CTM0 Comparator P match interrupt control
0: Disable
1: Enable

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	STMAF	STMPF	CTM1AF	CTM1PF	STMAE	STMPE	CTM1AE	CTM1PE
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 **STMAF**: STM Comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 6 **STMPF**: STM Comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 5 **CTM1AF**: CTM1 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 **CTM1PF**: CTM1 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 **STMAE**: STM Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **STMPE**: STM Comparator P match interrupt control
0: Disable
1: Enable

Bit 1 **CTM1AE**: CTM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **CTM1PE**: CTM1 Comparator P match interrupt control
0: Disable
1: Enable

• **MFI2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	SIMF	PTM0AF	PTM0PF	—	SIME	PTM0AE	PTM0PE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Umimplemented, read as “0”

Bit 6 **SIMF**: SIM interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 5 **PTM0AF**: PTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **PTM0PF**: PTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3 Umimplemented, read as "0"

Bit 2 **SIME**: SIM interrupt control
0: Disable
1: Enable

Bit 1 **PTM0AE**: PTM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTM0PE**: PTM0 Comparator P match interrupt control
0: Disable
1: Enable

• MFI3 Register

Bit	7	6	5	4	3	2	1	0
Name	PTM2AF	PTM2PF	PTM1AF	PTM1PF	PTM2AE	PTM2PE	PTM1AE	PTM1PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PTM2AF**: PTM2 Comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 6 **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 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 **PTM2AE**: PTM2 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **PTM2PE**: PTM2 Comparator P match 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

• **MFI4 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTM2AF	CTM2PF	DEF	LVF	CTM2AE	CTM2PE	DEE	LVE
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 **CTM2AF**: CTM2 Comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 6 **CTM2PF**: CTM2 Comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 5 **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 **LVF**: LVD interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3 **CTM2AE**: CTM2 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **CTM2PE**: CTM2 Comparator P match interrupt control
0: Disable
1: Enable

Bit 1 **DEE**: Data EEPROM interrupt control
0: Disable
1: Enable

Bit 0 **LVE**: LVD interrupt control
0: Disable
1: Enable

Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A match or A/D conversion completion etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a

new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a "JMP" which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a "RETI", which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the Interrupt Structure diagram shows the priority that is applied. All of the interrupt request flags when set will wake up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.

External Interrupts

The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when an active transition 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 bits, 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. 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.

When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pins, 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.

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.

Touch Key Interrupt

A Touch Key interrupt will take place when the Touch Key interrupt request flag, TMKF, is set, a situation that will occur when the time slot counter overflows in manual mode or all the touch key auto scan operations finish. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Touch Key interrupt enable bit, TKME, must be first set. When the interrupt is enabled, the stack is not full and the touch key time slot counter overflow occurs

in manual mode or all the touch key auto scan operations finish, a subroutine call to the relevant interrupt vector will take place. When the interrupt is serviced, the Touch Key interrupt request flag will be automatically reset and the EMI bit will also be automatically cleared to disable other interrupts.

UART Interrupts

The UARTn Interrupts are controlled by several UARTn transfer conditions. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RXn/TXn pin wake-up. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, and UARTn Interrupt enable bit, URnE, must first be set. When the interrupt is enabled, the stack is not full and any of the conditions described above occurs, a subroutine call to the corresponding UARTn Interrupt vector, will take place. When the interrupt is serviced, the UARTn Interrupt flag, URnF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the UnSR register flags will only be cleared when certain actions are taken by the UARTn, the details of which are given in the UART section.

I²C Interrupt

For the independent I²C interface, an I²C interrupt request will take place when the I²C Interrupt request flag, IICF, is set, which occurs when a byte of data has been received or transmitted by the I²C interface, or an I²C slave address match occurs, or an I²C bus time-out occurs. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the I²C Interrupt enable bit, IICE, 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 I²C Interrupt vector will take place. When the interrupt is serviced, the I²C Interrupt flag, IICF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

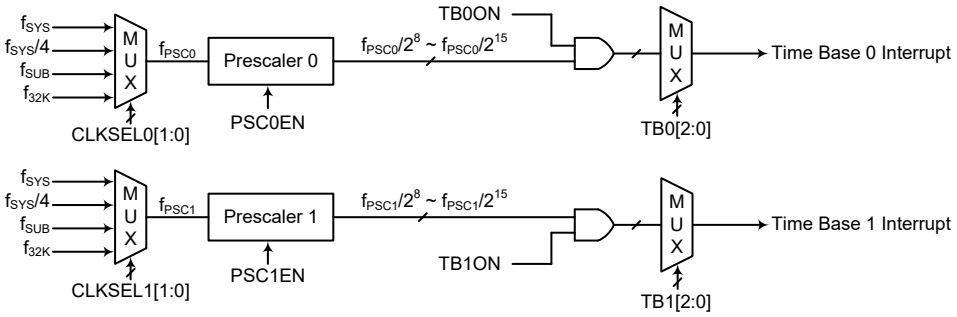
A/D Converter Interrupt

The A/D Converter interrupt is controlled by the termination of the A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when an A/D conversion process finishes in the manual triggered conversion mode, or when the specified N times of automatic A/D conversions have completed in the automatic triggered conversion mode. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process as described above has ended, a subroutine call to the A/D Converter Interrupt vector will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

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 flag, 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 bit, 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 Interrupts is to provide an interrupt signal at fixed time periods. Their respective clock source, f_{PSC0} or f_{PSC1}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$, f_{SUB} or f_{32K}, and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the PSC0R and PSC1R register respectively.



Time Base Interrupts

• PSCnR Register (n=0~1)

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PSCnEN	CLKSELn1	CLKSELn0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 7 **PSCnEN**: Prescaler n enable control
0: Disable
1: Enable

Bit 1~0 **CLKSELn1~CLKSELn0**: Prescaler n clock source f_{PSCn} selection
00: f_{SYS}
01: $f_{SYS}/4$
10: f_{SUB}
11: f_{32K}

• 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 enable control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TBn2~TBn0**: Time Base n time-out period selection
000: $2^8/f_{PSCn}$
001: $2^9/f_{PSCn}$
010: $2^{10}/f_{PSCn}$
011: $2^{11}/f_{PSCn}$
100: $2^{12}/f_{PSCn}$
101: $2^{13}/f_{PSCn}$
110: $2^{14}/f_{PSCn}$
111: $2^{15}/f_{PSCn}$

Multi-function Interrupts

Within the device there are five Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts, OVP interrupt, SIM interrupt, LVD interrupt and EEPROM interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags MFnF are set, which will occur 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 interrupt enable bits of the functions which are included in the multi-function interrupts, and relevant Multi-function Interrupt enable bit, MFnE, 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 MFnF will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

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

EEPROM Interrupt

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

LVD Interrupt

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

Over Voltage Protection Interrupt

The Over Voltage Protection Interrupt is also known as the OVP Interrupt. The OVP Interrupt is contained within the Multi-function Interrupt.The OVP Interrupt is controlled by detecting the OVPINP input voltage. An OVP Interrupt request will take place when the OVP Interrupt request flag, OVPF, is set, which occurs when the OVP circuit detects an over voltage condition. To allow the program to branch to the relevant interrupt vector address, the global interrupt enable bit, EMI, the OVP Interrupt enable bit, OVPE, and the relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and an over voltage condition is detected, a subroutine call to the relevant Multi-function Interrupt vector location will take place. When the OVP Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag MFnF will be also automatically cleared. As the OVP interrupt request flag will not be automatically cleared, they have to be cleared by the application program.

Serial Interface Module Interrupt

The Serial Interface Module Interrupt is also known as the SIM Interrupt. The SIM Interrupt is contained within the Multi-function Interrupt. A SIM Interrupt request will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface, an I^2C slave address match or I^2C bus time-out occurs. To allow the program to branch to the relevant interrupt vector address, the global interrupt enable bit, EMI, the SIM Interrupt enable bit, SIME, and the associated Multi-function interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the corresponding Multi-function Interrupt vector will take place. When the SIM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag MFnF will be also automatically cleared. As the SIMF flag will not be automatically cleared, it has to be cleared by the application program.

TM Interrupts

The Compact, Standard and Periodic Type TMs each have two interrupts and come from the comparator P or A match situation. All of these TM interrupts are contained within the Multi-function Interrupts. For each of the Compact, Standard and Periodic Type TMs there are two interrupt request flags and two interrupt 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 the associated 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 Multi-function interrupt request flag MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device

is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program.

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MFnF, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

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

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

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

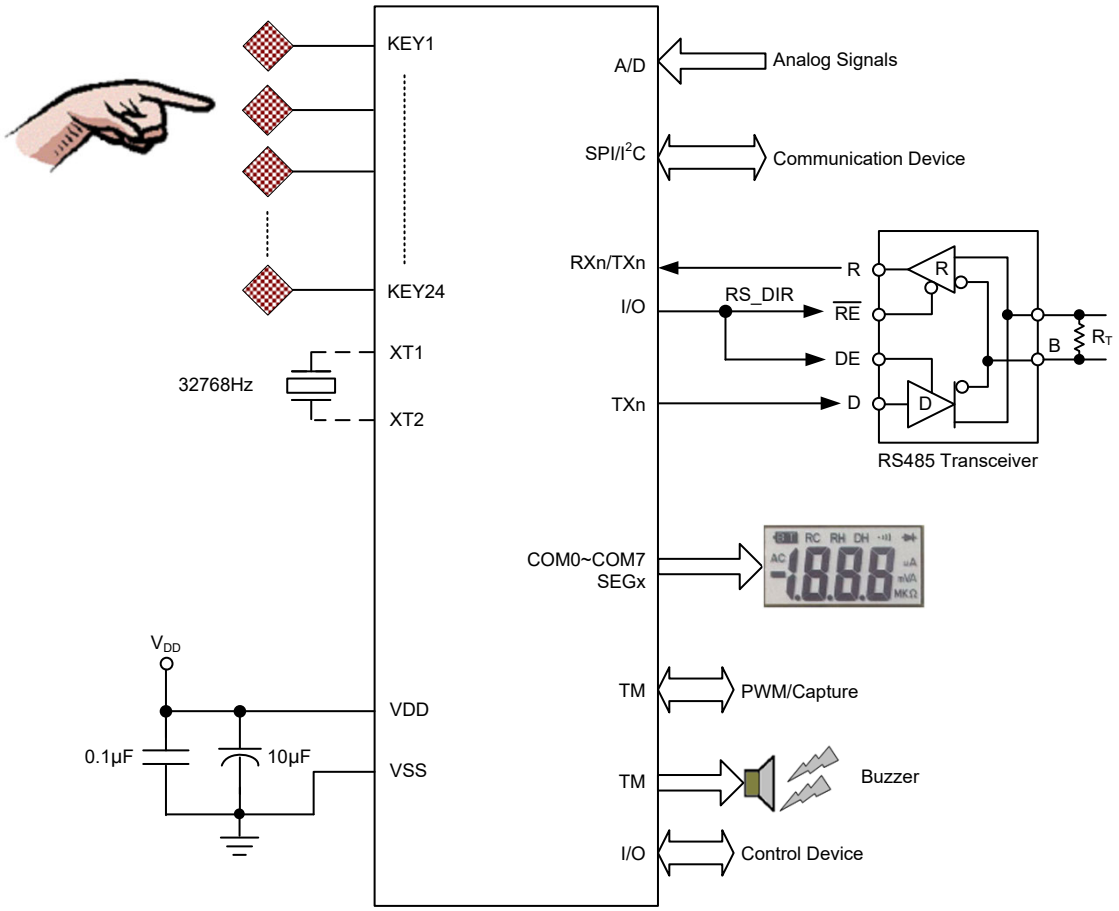
Configuration Options

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}: 8MHz, 12MHz or 16MHz

Note: When the HIRC has been configured at a frequency shown in this table, the HIRC1~HIRC0 bits should also be setup to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Application Circuits



Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs 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 can not 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

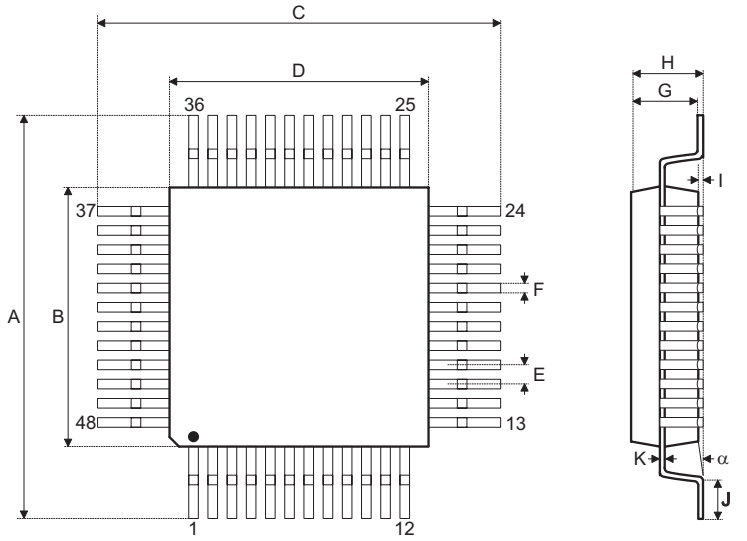
Package Information

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

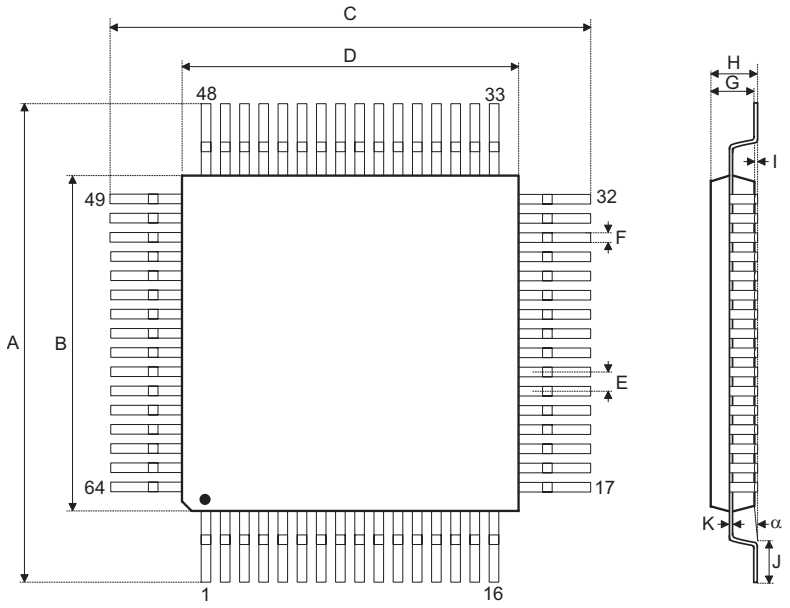
48-pin LQFP (7mm×7mm) Outline Dimensions



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	
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		9.00 BSC	
B		7.00 BSC	
C		9.00 BSC	
D		7.00 BSC	
E		0.50 BSC	
F	0.17	0.22	0.27
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

64-pin LQFP (7mm×7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.354 BSC	
B		0.276 BSC	
C		0.354 BSC	
D		0.276 BSC	
E		0.016 BSC	
F	0.005	0.007	0.009
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		9.00 BSC	
B		7.00 BSC	
C		9.00 BSC	
D		7.00 BSC	
E		0.40 BSC	
F	0.13	0.18	0.23
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

Copyright© 2026 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

The information provided in this document has been produced with reasonable care and attention before publication, however, HOLTEK does not guarantee that the information is completely accurate. The information contained in this publication is provided for reference only and may be superseded by updates. HOLTEK disclaims any expressed, implied or statutory warranties, including but not limited to suitability for commercialization, satisfactory quality, specifications, characteristics, functions, fitness for a particular purpose, and non-infringement of any third-party's rights. HOLTEK disclaims all liability arising from the information and its application. In addition, HOLTEK does not recommend the use of HOLTEK's products where there is a risk of personal hazard due to malfunction or other reasons. HOLTEK hereby declares that it does not authorise the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.