Glucose Meter LCD Flash MCU

BH67F2472

Revision: V1.51 Date: March 03, 2025

www.holtek.com

Features

CPU Features

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

Peripheral Features

- Flash Program Memory: 32K×16
- Data Memory: 2048×8
- True EEPROM Memory: 2048×8
- In Application Programming function – IAP
- On-Chip Debug Support function – OCDS
- Watchdog Timer function
- 58 bidirectional I/O lines
- Four external interrupt lines shared with I/O pins
- Multiple Timer Modules for time measure, compare match output, PWM output function or single pulse output function
- Dual Time-Base functions for generation of fixed time interrupt signals
- Glucose Meter AFE circuitry
  - 6 external channel 12-bit resolution A/D Converter
  - Internal Reference Voltage Generator
  - 12-bit D/A Converter
  - Two Operational Amplifiers

- LCD Driver function
  - SEGs × COMs: 36×4, 34×6 or 32×8
  - Duty type: 1/4 duty, 1/6 duty or 1/8 duty
  - Bias level: 1/3 bias
  - Bias type: C type
  - Waveform type: type A or type B
- Two Universial Serial Interface Modules – USIM for SPI, I^2C or UART communication
- Single Independent Serial Peripheral Interface – SPI
- Low voltage reset function
- Package types: 64/80-pin LQFP

Development Tools

For rapid product development and to simplify device parameter setting, Holtek has provided relevant development tools which users can download from the following link:

https://www.holtek.com/page/tool-detail/dev_plat/voice/voice_workshop

General Description

The device is a Flash Memory 8-bit high performance RISC architecture microcontroller device with Glucose meter AFE module and LCD driver that specifically designed for Glucose Meter with LCD display applications.

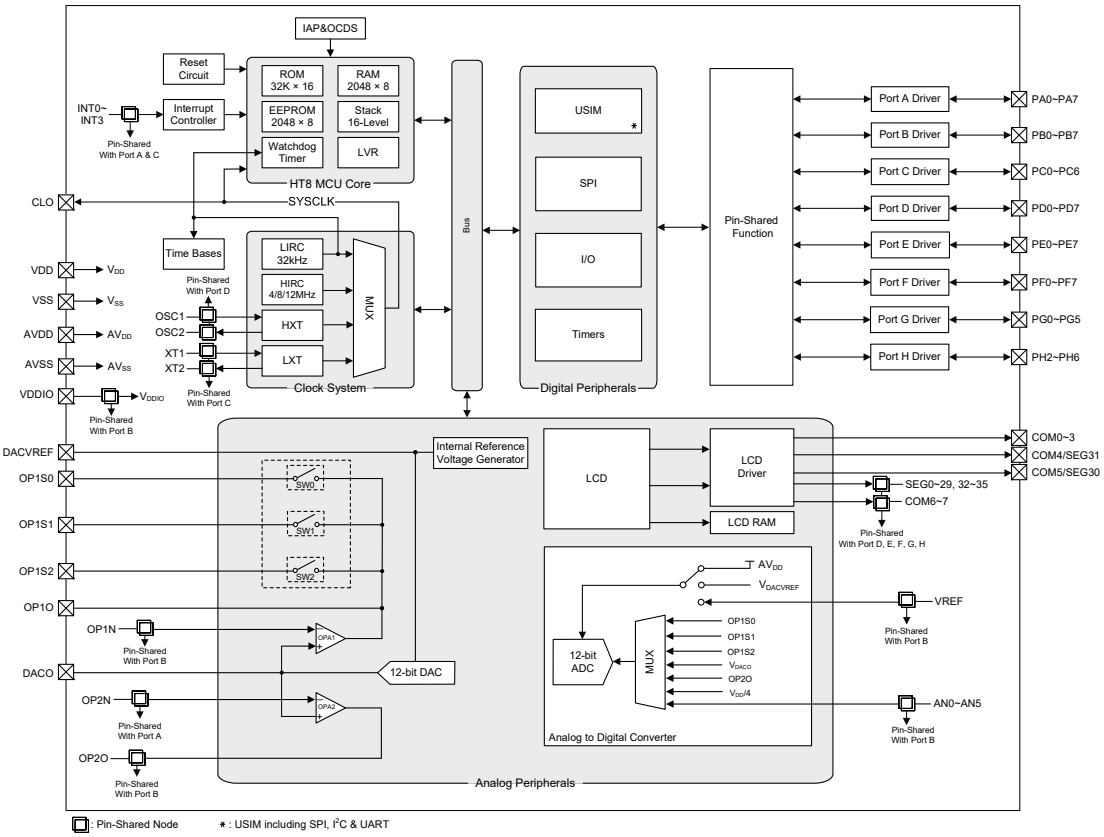
For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc. By using the In Application Programming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

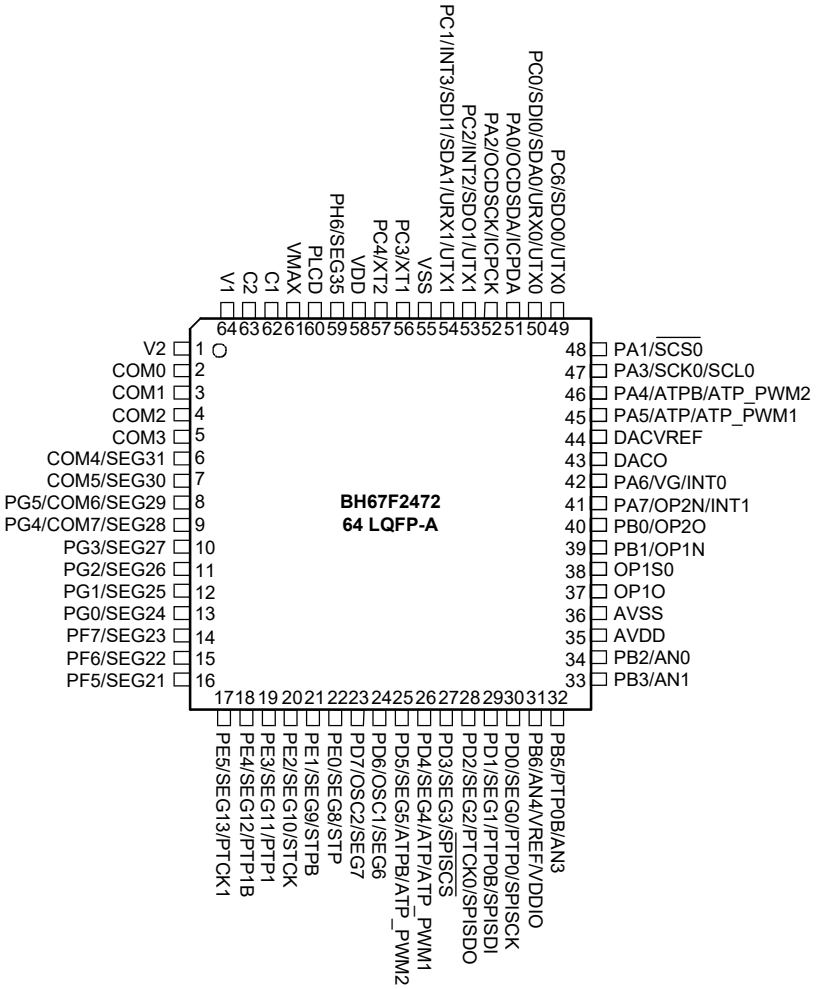
A full choice of external, internal high and low oscillators is provided including fully integrated system oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users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

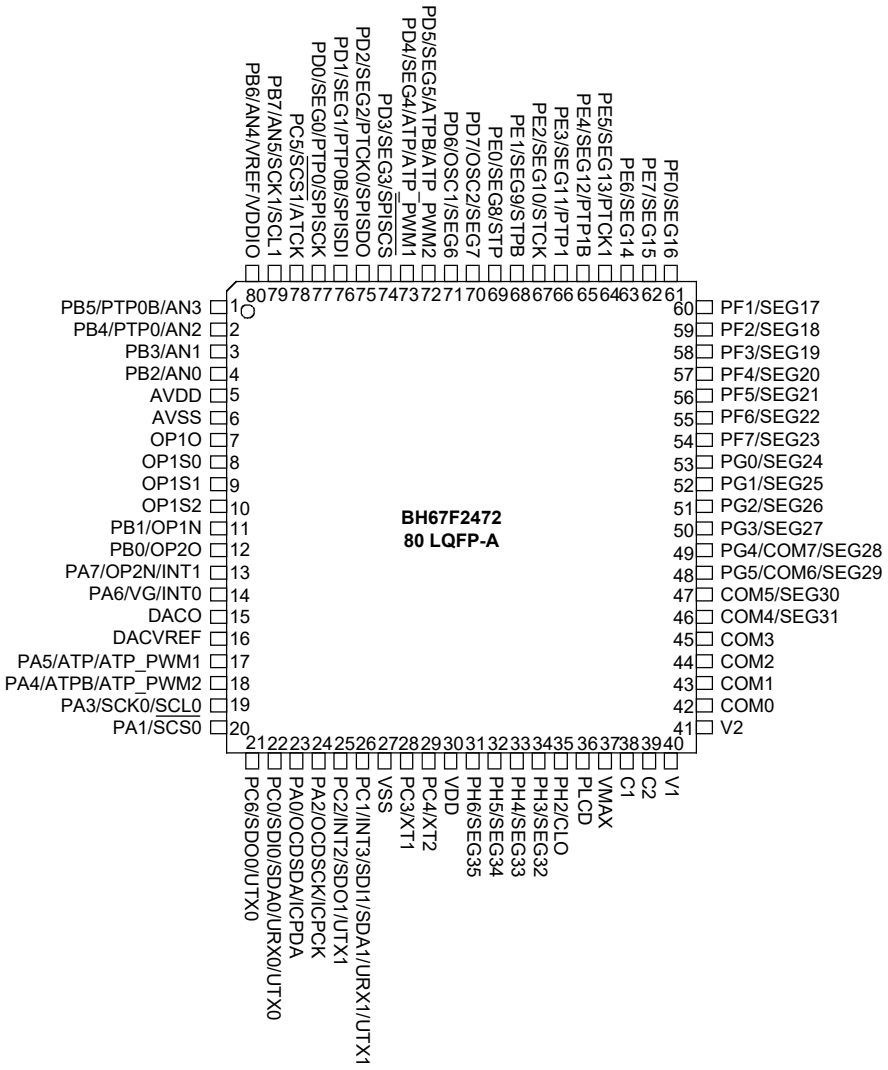
With regard to Glucose Meter applications, the device has integrated many of the functions required by these products. These include functions such as Internal Reference Voltage generator, 12-bit D/A Converter, LCD driver function, etc. The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the device will be highly capable of providing MCU solutions for Glucose Meter applications which require an LCD display.

Block Diagram



Pin Assignment





Note: 1. For a pin which has multiple pin-shared functions, the desired pin function is determined by the corresponding software control bits.

2. For the less pin count package type there will be unbounded pins which should be properly configured to avoid unwanted power consumption resulting from floating input conditions. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.

Pin Descriptions

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 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/OCDSDA/ICPDA	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OCDSDA	—	ST	CMOS	OCDS data/address pin
	ICPDA	—	ST	CMOS	ICP data/address pin
PA1/$\overline{SCS0}$	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	$\overline{SCS0}$	PAS0	ST	CMOS	USIM0 SPI slave select
PA2/OCDSCK/ICPCK	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OCDSCK	—	ST	—	OCDS clock pin
	ICPCK	—	ST	—	ICP clock pin
PA3/SCK0/SCL0	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SCK0	PAS0	ST	CMOS	USIM0 SPI serial clock
	SCL0	PAS0	ST	NMOS	USIM0 I^2C clock line
PA4/ATPB/ATP_PWM2	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	ATPB	PAS1	—	CMOS	ATM inverted output
	ATP_PWM2	PAS1	—	CMOS	ATM Audio PWM Mode output 2
PA5/ATP/ATP_PWM1	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	ATP	PAS1	—	CMOS	ATM output
	ATP_PWM1	PAS1	—	CMOS	ATM Audio PWM Mode output 1
PA6/VG/INT0	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	VG	PAS1	AN	—	Virtual Ground
	INT0	PAS1 INTEG INTC0	ST	—	External interrupt input
PA7/OP2N/INT1	PA7	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OP2N	PAS1	AN	—	OPA2 negative input
	INT1	PAS1 INTEG INTC0	ST	—	External interrupt input
PB0/OP2O	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	OP2O	PBS0	—	AN	OPA2 output
PB1/OP1N	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	OP1N	PBS0	AN	—	OPA1 negative input

Pin Name	Function	OPT	I/T	O/T	Description
PB2/AN0	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	AN0	PBS0	AN	—	A/D Converter external input channel
PB3/AN1	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	AN1	PBS0	AN	—	A/D Converter external input channel
PB4/PTP0/AN2	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP0	PBS1	—	CMOS	PTM0 output
	AN2	PBS1	AN	—	A/D Converter external input channel
PB5/PTP0B/AN3	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP0B	PBS1	—	CMOS	PTM0 inverted output
	AN3	PBS1	AN	—	A/D Converter external input channel
PB6/AN4/VREF/VDDIO	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	AN4	PBS1	AN	—	A/D Converter external input channel
	VREF	PBS1	AN	—	A/D Converter external reference voltage input
	VDDIO	PBS1	PWR	—	PD0~PD3 pin power supply
PB7/AN5/SCK1/SCL1	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SCK1	PBS1	ST	CMOS	USIM1 SPI serial clock
	SCL1	PBS1	ST	NMOS	USIM1 I²C clock line
	AN5	PBS1	AN	—	A/D Converter external input channel
PC0/SDI0/SDA0/URX0/ UTX0	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SDI0	PCS0	ST	—	USIM0 SPI serial data input
	SDA0	PCS0	ST	NMOS	USIM0 I²C data line
	URX0/ UTX0	PCS0	ST	CMOS	USIM0 UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication
PC1/INT3/SDI1/SDA1/ URX1/UTX1	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	INT3	PCS0 INTEG INTC2	ST	—	External interrupt input
	SDI1	PCS0	ST	—	USIM1 SPI serial data input
	SDA1	PCS0	ST	NMOS	USIM1 I²C data line
	URX1/ UTX1	PCS0	ST	CMOS	USIM1 UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication
PC2/INT2/SDO1/UTX1	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	INT2	PCS0 INTEG INTC1	ST	—	External interrupt input
	SDO1	PCS0	—	CMOS	USIM1 SPI serial data output
	UTX1	PCS0	—	CMOS	USIM1 UART serial data output
PC3/XT1	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	XT1	PCS0	LXT	—	LXT oscillator pin

Pin Name	Function	OPT	I/T	O/T	Description
PC4/XT2	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	XT2	PCS1	—	LXT	LXT oscillator pin
PC5/$\overline{SCS1}$/ATCK	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	$\overline{SCS1}$	PCS1	ST	CMOS	USIM1 SPI slave select
	ATCK	PCS1	ST	—	ATM clock input
PC6/SDO0/UTX0	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SDO0	PCS1	—	CMOS	USIM0 SPI serial data output
	UTX0	PCS1	—	CMOS	USIM0 UART serial data output
PD0/SEG0/PTP0/ SPISCK	PD0	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG0	PDS0	—	AN	LCD Segment output
	PTP0	PDS0	—	CMOS	PTM0 output
	SPISCK	PDS0	ST	CMOS	SPI serial clock
PD1/SEG1/PTP0B/ SPISDI	PD1	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG1	PDS0	—	AN	LCD Segment output
	PTP0B	PDS0	—	CMOS	PTM0 inverted output
	SPISDI	PDS0	ST	—	SPI serial data input
PD2/SEG2/PTCK0/ SPISDO	PD2	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG2	PDS0	—	AN	LCD Segment output
	PTCK0	PDS0	ST	—	PTM0 clock input
	SPISDO	PDS0	—	CMOS	SPI serial data output
PD3/SEG3/$\overline{SPISCS}$	PD3	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG3	PDS0	—	AN	LCD Segment output
	$\overline{SPISCS}$	PDS0	ST	CMOS	SPI slave select
PD4/SEG4/ATP/ ATP_PWM1	PD4	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG4	PDS1	—	AN	LCD Segment output
	ATP	PDS1	—	CMOS	ATM output
	ATP_PWM1	PDS1	—	CMOS	ATM Audio PWM Mode output 1
PD5/SEG5/ATPB/ ATP_PWM2	PD5	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG5	PDS1	—	AN	LCD Segment output
	ATPB	PDS1	—	CMOS	ATM inverted output
	ATP_PWM2	PDS1	—	CMOS	ATM Audio PWM Mode output 2
PD6/OSC1/SEG6	PD6	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	OSC1	PDS1	HXT	—	HXT oscillator pin
	SEG6	PDS1	—	AN	LCD Segment output
PD7/OSC2/SEG7	PD7	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	OSC2	PDS1	—	HXT	HXT oscillator pin
	SEG7	PDS1	—	AN	LCD Segment output

Pin Name	Function	OPT	I/T	O/T	Description
PE0/SEG8/STP	PE0	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG8	PES0	—	AN	LCD Segment output
	STP	PES0	—	CMOS	STM output
PE1/SEG9/STPB	PE1	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG9	PES0	—	AN	LCD Segment output
	STPB	PES0	—	CMOS	STM inverted output
PE2/SEG10/STCK	PE2	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG10	PES0	—	AN	LCD Segment output
	STCK	PES0	ST	—	STM clock input
PE3/SEG11/PTP1	PE3	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG11	PES0	—	AN	LCD Segment output
	PTP1	PES0	—	CMOS	PTM1 output
PE4/SEG12/PTP1B	PE4	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG12	PES1	—	AN	LCD Segment output
	PTP1B	PES1	—	CMOS	PTM1 inverted output
PE5/SEG13/PTCK1	PE5	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG13	PES1	—	AN	LCD Segment output
	PTCK1	PES1	ST	—	PTM1 clock input
PE6/SEG14	PE6	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG14	PES1	—	AN	LCD Segment output
PE7/SEG15	PE7	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG15	PES1	—	AN	LCD Segment output
PF0/SEG16~PF3/ SEG19	PF0~3	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG16~19	PFS0	—	AN	LCD Segment output
PF4/SEG20~PF7/ SEG23	PF4~7	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG20~23	PFS1	—	AN	LCD Segment output
PG0/SEG24~PG3/ SEG27	PG0~3	PGPU PGS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG24~27	PGS0	—	AN	LCD Segment output
PG4/COM7/SEG28	PG4	PGPU PGS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	COM7	PGS1	—	AN	LCD Common output
	SEG28	PGS1	—	AN	LCD Segment output
PG5/COM6/SEG29	PG5	PGPU PGS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	COM6	PGS1	—	AN	LCD Common output
	SEG29	PGS1	—	AN	LCD Segment output
PH2/CLO	PH2	PHPU PHS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	CLO	PHS0	—	CMOS	System clock output

Pin Name	Function	OPT	I/T	O/T	Description
PH3/SEG32	PH3	PHPU PHS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG32	PHS0	—	AN	LCD Segment output
PH4/SEG33~PH6/ SEG35	PH4~6	PHPU PHS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG33~35	PHS1	—	AN	LCD Segment output
COM0~COM3	COM0~3	—	—	AN	LCD Common output
COM4/SEG31	COM4	COMS	—	AN	LCD Common output
	SEG31	COMS	—	AN	LCD Segment output
COM5/SEG30	COM5	COMS	—	AN	LCD Common output
	SEG30	COMS	—	AN	LCD Segment output
VMAX	VMAX	—	PWR	—	LCD maximum voltage, should be connected to VDD or V1
PLCD	PLCD	—	PWR	AN	LCD power supply
V1	V1	—	PWR	AN	LCD voltage pump
V2	V2	—	PWR	AN	LCD voltage pump
C1	V1	—	PWR	AN	LCD voltage pump
C2	V2	—	PWR	AN	LCD voltage pump
OP1S0	OP1S0	—	—	AN	OPA1 output 0 controlled by analog switch
OP1S1	OP1S1	—	—	AN	OPA1 output 1 controlled by analog switch
OP1S2	OP1S2	—	—	AN	OPA1 output 2 controlled by analog switch
OP1O	OP1O	—	—	AN	OPA1 output
DACVREF	DACVREF	—	—	AN	D/A Converter reference voltage
DACO	DACO	—	—	AN	D/A Converter output
VDD	VDD	—	PWR	—	Digital positive power supply
VSS	VSS	—	PWR	—	Digital negative power supply
AVDD	AVDD	—	PWR	—	Analog positive power supply
AVSS	AVSS	—	PWR	—	Analog negative power supply

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

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}-0.3V to 6.0V

Input Voltage ..... V_{SS}-0.3V to V_{DD}+0.3V

Storage Temperature..... -50°C to 125°C

Operating Temperature..... -40°C to 85°C

I_{OH} Total..... -80mA

I_{OL} Total 80mA

Total Power Dissipation 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the 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}=4MHz	2.2	—	5.5	V
		f_{SYS}=8MHz	2.2	—	5.5	
		f_{SYS}=12MHz	2.7	—	5.5	
	Operating Voltage – HXT	f_{SYS}=4MHz	2.2	—	5.5	V
		f_{SYS}=8MHz	2.2	—	5.5	
		f_{SYS}=12MHz	2.7	—	5.5	
		f_{SYS}=16MHz	3.3	—	5.5	
	Operating Voltage – LXT	f_{SYS}=32.768kHz	2.2	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32kHz	2.2	—	5.5	V

Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2.2V	f_{SYS}=32kHz	—	15	30	μA
		3V		—	18	36	
		5V		—	30	50	
	SLOW Mode – LXT	2.2V	f_{SYS}=32.768kHz	—	15	30	μA
		3V		—	18	36	
		5V		—	30	50	

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	FAST Mode – HIRC	2.2V	f_{SYS}=4MHz	—	0.3	0.5	mA
		3V		—	0.4	0.6	
		5V		—	0.8	1.2	
		2.2V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	
		2.7V	f_{SYS}=12MHz	—	1.0	1.4	mA
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	
	FAST Mode – HXT	2.2V	f_{SYS}=4MHz	—	200	500	μA
		3V		—	250	800	
		5V		—	500	1300	
		2.2V	f_{SYS}=8MHz	—	320	700	μA
		3V		—	500	1000	
		5V		—	1000	1600	
		2.7V	f_{SYS}=12MHz	—	0.7	1.2	mA
		3V		—	0.75	1.60	
		5V		—	2.2	3.0	
		3.3V	f_{SYS}=16MHz	—	1.5	3.0	mA
		5V		—	2.5	5.0	

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

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

Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.2V	WDT off	—	0.8	1.2	10.0	μA
		3V		—	0.8	1.2	10.0	
		5V		—	1.0	1.5	12.0	
		2.2V	WDT on	—	1.2	2.4	10.0	μA
		3V		—	1.5	3.0	12.0	
		5V		—	3	5	15	
	IDLE0 Mode– LIRC	2.2V	f_{SUB} on	—	2.4	4.0	12.0	μA
		3V		—	3	5	15	
		5V		—	5	10	18	
	IDLE0 Mode– LXT	2.2V	f_{SUB} on	—	2.4	4.0	12.0	μA
		3V		—	3	5	15	
		5V		—	5	10	18	

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	IDLE1 Mode – HIRC	2.2V	f_{SUB} on, f_{SYS}=4MHz	—	144	200	240	μA
		3V		—	180	250	300	
		5V		—	400	600	720	
		2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	540	750	900	
		5V		—	800	1200	1440	
	IDLE1 Mode – HXT	2.2V	f_{SUB} on, f_{SYS}=4MHz	—	144	200	240	μA
		3V		—	180	250	300	
		5V		—	600	800	960	
		2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	800	1200	1440	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	540	750	900	
		5V		—	1200	2000	2400	
		3.3V	f_{SUB} on, f_{SYS}=16MHz	—	1.1	1.6	1.9	mA
		5V		—	1.6	2.4	3.0	

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

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

A.C. Characteristics

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

Internal High Speed Oscillator – HIRC – Frequency Accuracy

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

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

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

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

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

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

Internal Low Speed Oscillator Characteristics – LIRC

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

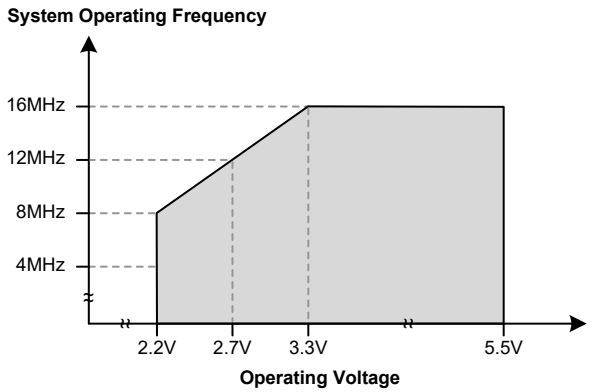
External Low Speed Crystal Oscillator Characteristics – LXT

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{LXT}	LXT Frequency	2.2V~5.5V	—	—	32768	—	Hz
Duty Cycle	Duty Cycle	—	—	40	—	60	%
t_{START}	LXT Start Up Time	3V	—	—	—	1000	ms
		5V	—	—	—	1000	
R_{NEG}	Negative Resistance	2.2V	—	3×ESR	—	—	Ω

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

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time (Wake-up from Conditions where f_{SYS} is off)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HXT}$	—	128	—	t_{HXT}
			$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
			$f_{SYS}=f_{SUB}=f_{LXT}$	—	1024	—	t_{LXT}
			$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time (Wake-up from Conditions where f_{SYS} is on)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HXT}$ or f_{HIRC}	—	2	—	t_H
			$f_{SYS}=f_{SUB}=f_{LXT}$ or f_{LIRC}	—	2	—	t_{SUB}
	System Speed Switch Time (FAST to Slow Mode or SLOW to FAST Mode)	—	f_{HXT} switches from off → on	—	1024	—	t_{HXT}
			f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
			f_{LXT} switches from off → on	—	1024	—	t_{LXT}
t_{RSTD}	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR_{POR}=5V/ms	14	16	18	ms
	System Reset Delay Time (LVRC/WDTC/RSTC Software Reset)	—	—				
	System Reset Delay Time (Reset Source from WDT Overflow)	—	—	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

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 (Except PD0~PD3 Pins)	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports (Except PD0~PD3 Pins)	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Ports (Except PD0~PD3 Pins)	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Ports (Except PD0~PD3 Pins)	3V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	
R_{PH}	Pull-high Resistance for I/O Ports (Note) (Except PD0~PD3 Pins)	3V	—	20	60	100	kΩ
		5V		10	30	50	
I_{LEAK}	Input leakage current for I/O Ports (Except PD0~PD3 Pins)	3V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
		5V					

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{TCK}	STCK, PTCKn, ATCK Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	External Interrupt Input Minimum Pulse Width	—	—	10	—	—	μs

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

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

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	V_{DD} Power Supply for PD0~PD3 Pins	—	—	2.2	5.0	5.5	V
V_{DDIO}	V_{DDIO} Power Supply for PD0~PD3 Pins	—	—	2.2	—	V_{DD}	V
V_{IL}	Input Low Voltage for PD0~PD3 Pins	5V	Pin power=V_{DD} or V_{DDIO} V_{DDIO}=V_{DD}	0	—	1.5	V
		—	Pin power=V_{DD} or V_{DDIO}	0	—	0.2(V_{DD}/V_{DDIO})	
V_{IH}	Input High Voltage for PD0~PD3 Pins	5V	Pin power=V_{DD} or V_{DDIO} V_{DDIO}=V_{DD}	3.5	—	5.0	V
		—	Pin power=V_{DD} or V_{DDIO}	0.8(V_{DD}/V_{DDIO})	—	V_{DD}/V_{DDIO}	
I_{OL}	Sink Current for PD0~PD3 Pins	3V	V_{OL}=0.1 (V_{DD} or V_{DDIO}) V_{DDIO}=V_{DD}	16	32	—	mA
		5V		32	65	—	mA
		5V	V_{OL}=0.1 (V_{DD} or V_{DDIO}) V_{DDIO}=3V	20	40	—	mA
I_{OH}	Source Current for PD0~PD3 Pins	3V	V_{OH}=0.9 (V_{DD} or V_{DDIO}) V_{DDIO}=V_{DD}	-4	-8	—	mA
		5V		-8	-16	—	mA
		5V	V_{OH}=0.9 (V_{DD} or V_{DDIO}) V_{DDIO}=3V	-2.5	-5.0	—	mA
R_{PH}	Pull-high Resistance for PD0~PD3 Pins (Note)	3V	Pin power=V_{DD} or V_{DDIO} V_{DDIO}=V_{DD}	20	60	100	kΩ
		5V		10	30	50	kΩ
		5V	Pin power=V_{DD} or V_{DDIO} V_{DDIO}=3V	36	110	180	kΩ
I_{LEAK}	Input Leakage Current for PD0~PD3 Pins	5V	V_{IN}=V_{SS} or V_{IN}=V_{DD} or V_{DDIO}	—	—	±1	μA

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

Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{RW}	V_{DD} for Read/Write	—	—	V_{DDmin}	—	V_{DDmax}	V
Flash Program Memory							
t_{FER}	IAP Erase Time	—	FWERTS=0	—	3.2	3.9	ms
		—	FWERTS=1	—	3.7	4.5	
t_{FWR}	IAP Write Time	—	FWERTS=0	—	2.2	2.7	ms
		—	FWERTS=1	—	3.0	3.6	
E_P	Cell Endurance	—	—	10K	—	—	E/W
t_{RETD}	ROM 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_{EEER}	Erase Time	—	EWERTS=0	—	3.2	3.9	ms
		—	EWERTS=1	—	3.7	4.5	
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	
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	—	Device in SLEEP Mode	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 Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	-3%	2.1	+3%	V
		—	LVR enable, voltage select 2.55V		2.55		
		—	LVR enable, voltage select 3.15V		3.15		
		—	LVR enable, voltage select 3.8V		3.8		
I_{LVROP}	LVR Operating Current	3V	LVR enable, V_{LVR}=2.1V	—	—	10	μA
		5V		—	10	15	
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
I_{LVR}	Additional Current for LVR Enable	5V	—	—	—	14	μA

Analog Front End Circuit Characteristics

Operational Amplifier Electrical Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OPA}	Operating Current	3V	V_P=V_N=1/2 V_{DD}	—	—	650	μA
A_{OL}	Open Loop Gain	3V	—	80	100	—	dB
Ro	Output Resistance	2.4V~3.6V	Ta=0°C~50°C, R_{LOAD}=50kΩ 0.2V < V_{OP} < V_{DD} - 1.4V (Voltage Follower Configuration)	—	—	260	Ω
I_{OS}	Input Offset Current	2.4V~3.6V	Ta=0°C~50°C	—	±5	—	nA
TC	Temperature Coefficient of Offset Voltage	3V	Ta=0°C~50°C	—	—	±20	μV/°C
GBW	Gain Bandwidth	3V	Ta=25°C, R_L=1MΩ, C_L=60pF, V_{IN}=V_{CM}/2	100	—	—	kHz
V_{OS}	Input Offset Voltage	3V	Without calibration	-15	—	15	mV
V_{CM}	OPAMP Common Mode Voltage Range	3V	—	0.1	—	V_{DD} - 1.4	V
V_{OR}	OPAMP Maximum Output Voltage Range	3V	—	V_{SS} + 0.1	—	V_{DD} - 0.1	V

Internal Reference Voltage Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IREF}	Internal Reference Voltage	3V	IREFEN=1, PVREF=10000000B	-3%	2.0	+3%	V
I_{IREF}	Additional Current for Internal Reference Voltage Enable	—	IREFEN=1	—	650	1000	μA
TC_{IREF}	Temperature Coefficient of Internal Reference Voltage	3V	Ta=10°C~40°C	—	±20	±40	ppm/°C

Analog Switch Electrical Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
R_{OP}	On Resistance (OP1S0/OP1S1/OP1S2)	3V	Ta=10°C~40°C (Isw=200μA)	—	—	1300	Ω
R_{VG}	On Resistance (VG)	3V	Ta=10°C~40°C, GSW3=1	—	—	30	Ω
I_{LEAK}	Leakage Current (VG)	3V	Ta=10°C~40°C	—	±0.5	—	nA

12-bit D/A Converter Electrical Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
DNL	Differential Non-linearity	3V	Ta=25°C, DACVRS[1:0]=00B	—	—	±8	LSB
INL	Integral Non-linearity	3V	Ta=25°C, DACVRS[1:0]=00B	—	—	±20	LSB
V_{DACO}	Output Voltage Range	—	—	0.00	—	1.00	1/2 $V_{DACVREF}$
V_{DACO_RIPPLE}	Output Voltage Ripple	3V	DACVRS[1:0]=10B DACO with 10μF Cap	-0.6	—	+0.6	mV

A/D Converter Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	2	—	AV_{DD}	V
DNL	Differential Non-linearity	3V	$V_{REF}=AV_{DD}$, $t_{ADCK}=0.5\mu s$	—	—	±3	LSB
		5V					
		3V	$V_{REF}=AV_{DD}$, $t_{ADCK}=10\mu s$				
		5V					
INL	Integral Non-linearity	3V	$V_{REF}=AV_{DD}$, $t_{ADCK}=0.5\mu s$	—	—	±4	LSB
		5V					
		3V	$V_{REF}=AV_{DD}$, $t_{ADCK}=10\mu s$				
		5V					
I_{ADC}	Additional Current for ADC Enable	3V	No load, $t_{ADCK}=0.5\mu s$	—	1	2	mA
		5V		—	1.5	3.0	
t_{ADCK}	Clock Period	—	—	0.5	—	10.0	μs
t_{ON2ST}	ADC On-to-Start Time	—	—	4	—	—	μs
t_{ADS}	Sampling Time	—	—	—	4	—	t_{ADCK}
t_{ADC}	Conversion time (Include ADC sample and hold time)	—	—	—	16	—	t_{ADCK}

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	—	C type, power supply from PLCD pin	2.0	—	3.7	V
		—	C type, power supply from V1 pin	3.0	—	5.5	V
		—	C type, power supply from V2 pin	1.0	—	1.8	V
		—	C type, power supply from V_A	3.0	—	5.5	V
		3.3V~5.5V	C type, power supply from V_B	-10%	3.0	+10%	V
		2.2V~5.5V	C type, power supply from V_C	-10%	1.04	+10%	V
I_{LCD}	Additional Current for LCD Enable (C type)	3V	No load, V_A=V1=V_{DD}, 1/3 bias	—	10	15	μA
		5V		—	13.5	20.0	
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	—	

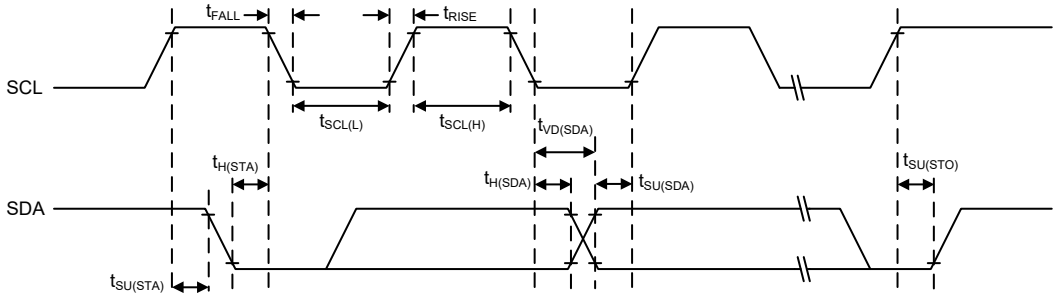
I²C Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{I2C}	I²C Standard Mode (100kHz) f_{SYS} Frequency (Note)	—	No clock debounce	2	—	—	MHz
			2 system clock debounce	4	—	—	
			4 system clock debounce	4	—	—	
	I²C Fast Mode (400kHz) f_{SYS} Frequency (Note)	—	No clock debounce	4	—	—	MHz
			2 system clock debounce	8	—	—	
			4 system clock debounce	8	—	—	
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	—	0.6	—	—	μs

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
$t_{SU(STO)}$	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.

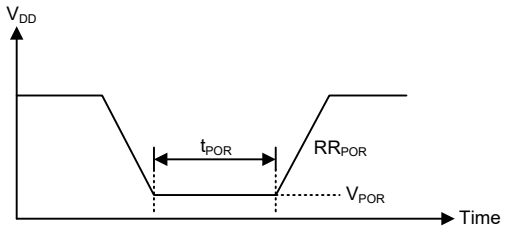


I²C Timing Diagram

Power-on Reset Characteristics

Ta=-40°C~85°C

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



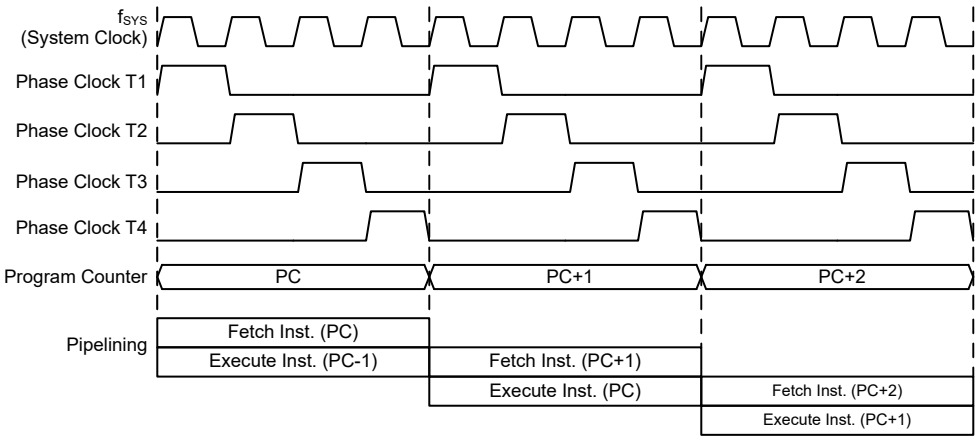
System Architecture

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

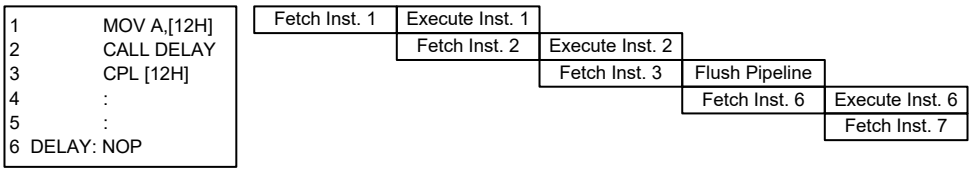
Clocking and Pipelining

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

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



System Clocking and Pipelining



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that 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 address may be located in a certain program memory bank which is selected by the program memory bank pointer bits, PBP1~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)**
PBP1~PBP0, PC12~PC8	PCL7~PCL0

Program Counter

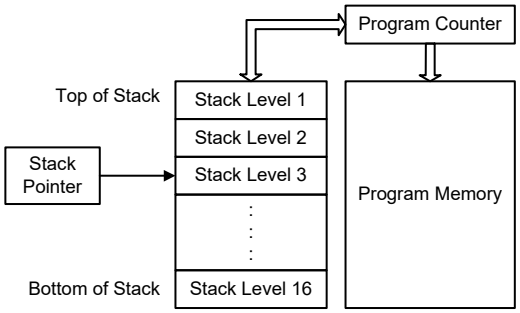
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

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

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

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



Arithmetic and Logic Unit – ALU

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

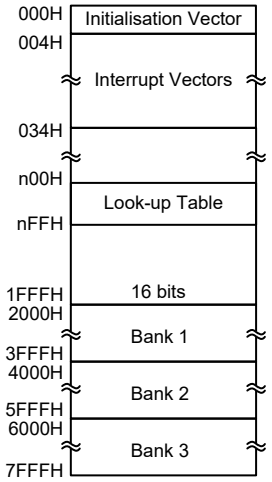
- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
  LAND, LOR, LXOR, LANDM, LORM, LXORM, LCPL, LCPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,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

Flash Program Memory

The Program Memory is the location where the user code or program is stored. For this device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, the Flash device 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 32K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer registers.



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

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

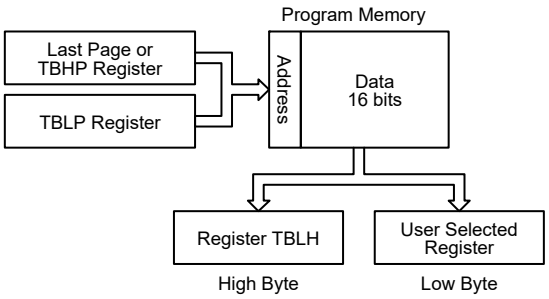


Table Program Example

The 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 3 and refers to the start address of the last page within the 32K 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 "7F06H" 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

```
rombank3 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,7fh             ; 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 "7F06H" 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 "7F05H" 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
```

```
:
:
code3 .section 'code'
org 1F00h          ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

In Circuit Programming – ICP

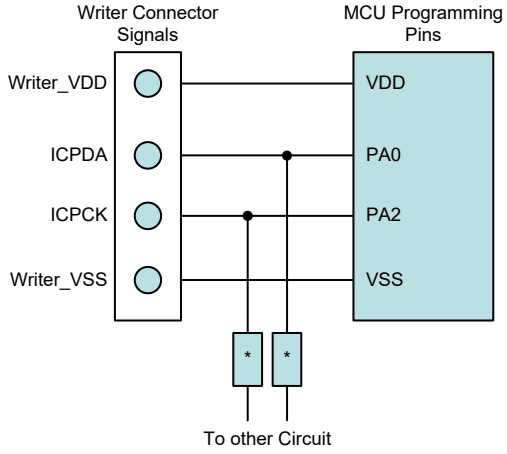
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides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

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

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

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

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



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

On-Chip Debug Support – OCDS

The device also provides the "On-Chip Debug" function for debuging during development process. Users can use the OCDS function to emulate the device behaviors 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 OCDS function for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect. 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	MCU 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 or USB, using I/O pins. Regarding the internal firmware, the user can select versions provided by Holtek or create their own. The following section illustrates the procedures regarding how to implement the IAP firmware.

Flash Memory Read/Write Size

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

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

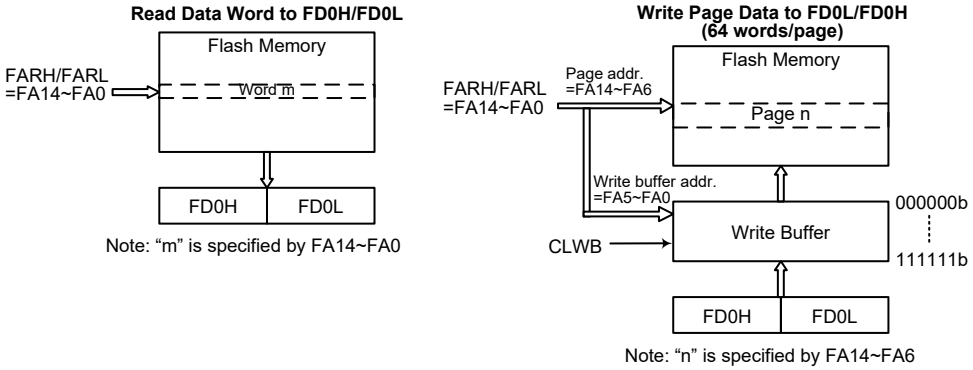
The Read operation can be carried out by executing a specific read procedure. The FRDEN bit is used to enable the read function and the FRD bit is used to initiate the read process by application programs and then indicate the read operation status. When the read process is finished, this bit will be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

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

IAP Operation Format

Page	FARH	FARL[7:6]	FARL[5:0]
0	0000 0000	00	Tag Address
1	0000 0000	01	
2	0000 0000	10	
3	0000 0000	11	
4	0000 0001	00	
: :	: :	: :	
510	0111 1111	10	
511	0111 1111	11	

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 64 words corresponding to a page. The write buffer address is mapped to a specific Flash memory page specified by the memory address bits, FA14~FA6. The data written into the FD0L and FD0H registers will be loaded into the write buffer. When data is written into the high byte data register, FD0H, it will result in the data stored in the high and low byte data registers both being written into the write buffer. It will also cause the Flash memory address to be incremented by one, after which the new address will be loaded into the FARH and FARL address registers. When the Flash memory address reaches the page boundary, 111111b of a page with 64 words, the address will now not be incremented but stop at the last address of the page. At this point a new page address should be specified for any other erase/write operations.

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

IAP Flash Program Memory Registers

There are two address registers, four pairs of 16-bit data registers and three control registers. 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 registers. 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, where n is equal to 0~3, and the control registers are named FC0, FC1 and FC2. As all the IAP related registers are located in Sector 1, they can be addressed directly only using the corresponding extended instructions or can be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pairs and Indirect Addressing Register, IAR1 or IAR2.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	FWERTS	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	FA14	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 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
011: Read Mode
110: Flash memory Erase/Write function Enable Mode
Other values: 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 control
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	—	FA14	FA13	FA12	FA11	FA10	FA9	FA8
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 6~0 **FA14~FA8**: Flash Memory Address bit 14 ~ 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

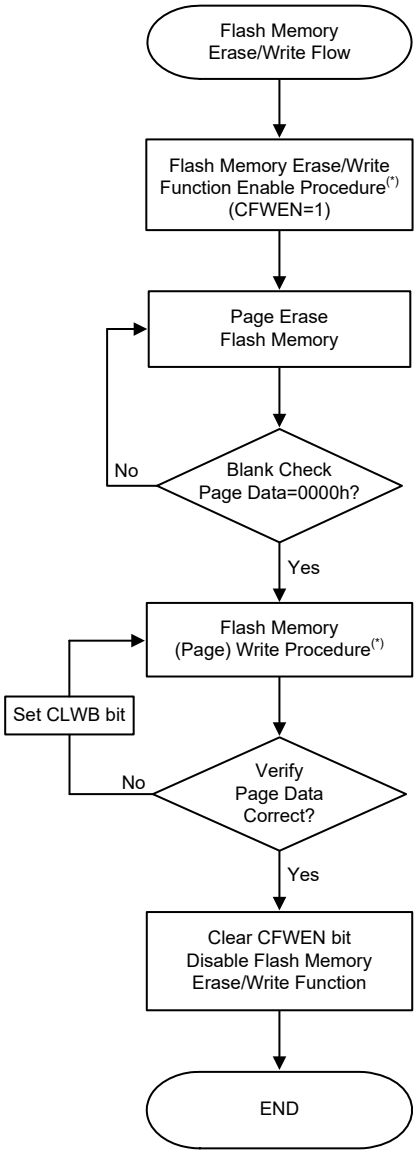
Flash Memory Erase/Write Flow

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, 111111b, 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

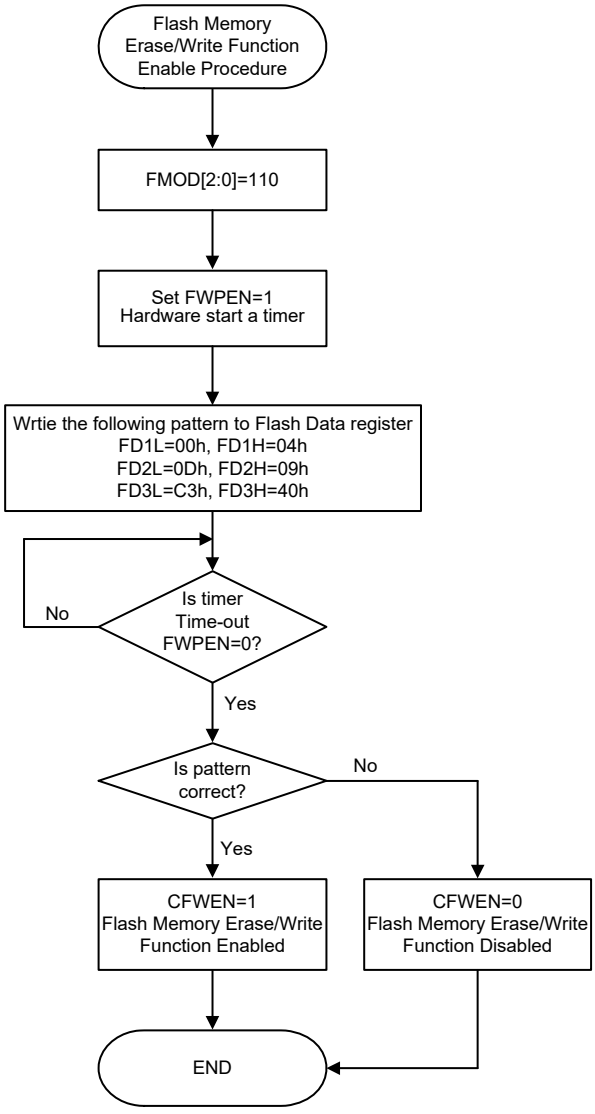
The Flash Memory Erase/Write Function Enable Mode is specially designed to prevent the Flash memory contents from being wrongly modified. In order to allow users to change the Flash memory data using the IAP control registers, users must first enable the Flash memory Erase/Write function.

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 of FD1L, FD2L, FD3L, FD1H, FD2H and FD3H, successively and as soon as possible after the FWPEN bit is set high. The enable Flash memory erase/write function data pattern is 00H, 0DH, C3H, 04H, 09H and 40H corresponding to the FD1L~FD3L and FD1H~FD3H registers respectively.
4. Once the timer has timed out, the FWPEN bit will automatically be cleared to 0 by hardware regardless of the input data pattern.
5. If the written data pattern is incorrect, the Flash memory erase/write function will not be enabled successfully and the above steps should be repeated. If the written data pattern is correct, the Flash memory erase/write function will be enabled successfully.
6. Once the Flash memory erase/write function is enabled, the Flash memory contents can be updated by executing the page erase and write operations using the IAP control registers.

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



Flash Memory Erase/Write Function Enable Procedure

Flash Memory Write Procedure

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

The write buffer size is 64 words, known as a page, whose address is mapped to a specific Flash memory page specified by the memory address bits, FA14~FA6. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits, FA14~FA6, specify.

Flash Memory Consecutive Write Description

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

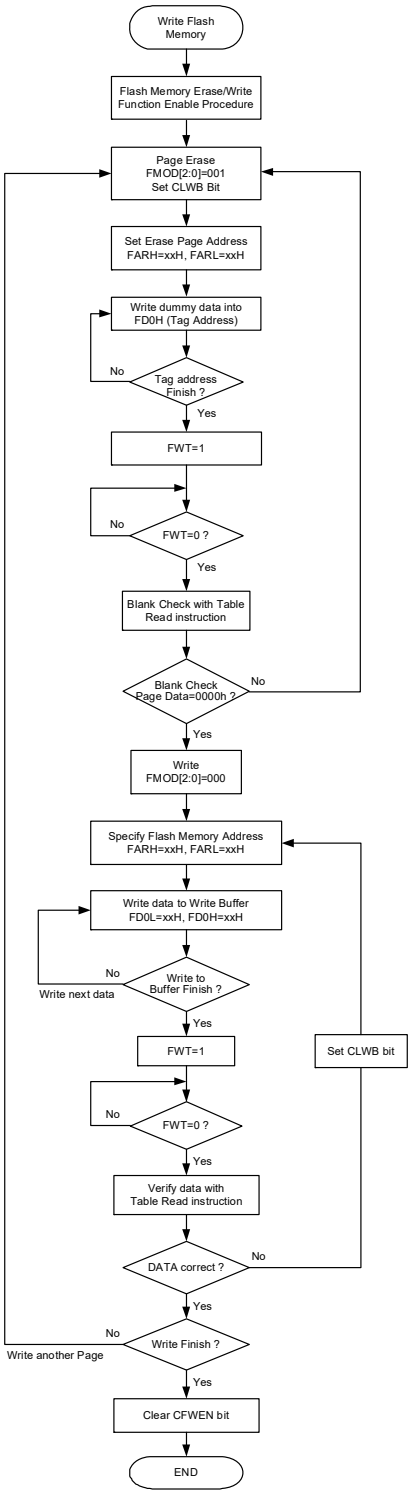
1. Activate the “Flash Memory Erase/Write function enable procedure”. Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the “Flash Memory Erase/Write function enable procedure” for more details.
2. Set the FMOD field to “001” to select the erase operation 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 64 words.
6. Set the FWT bit high to write the data words from the write buffer to the Flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

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

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



Flash Memory Consecutive Write Procedure

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

2. It will take 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.

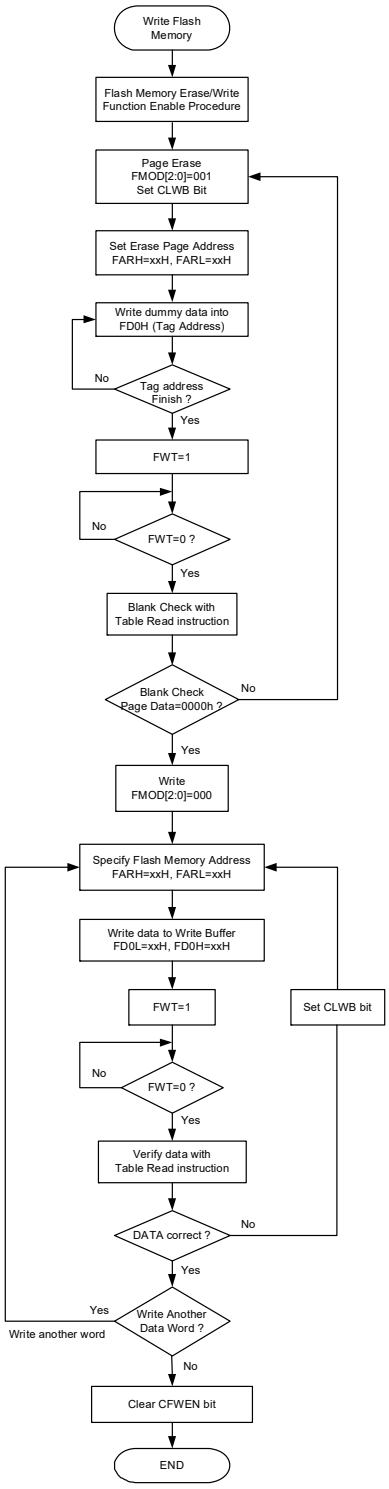
   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to "000" to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
6. Set the FWT bit high to 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.

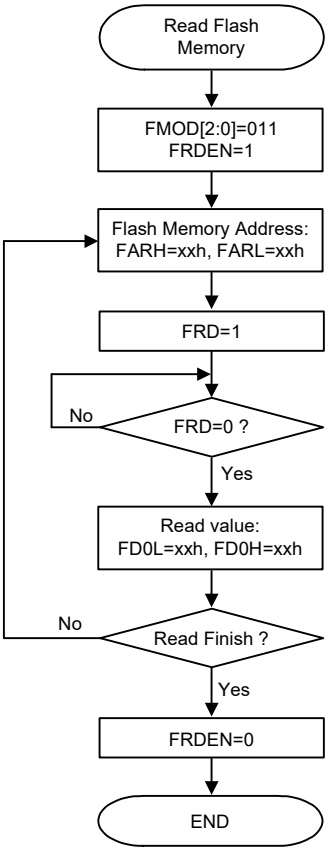
2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

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 an 8-bit wide RAM internal memory and is the location where temporary information is stored.

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

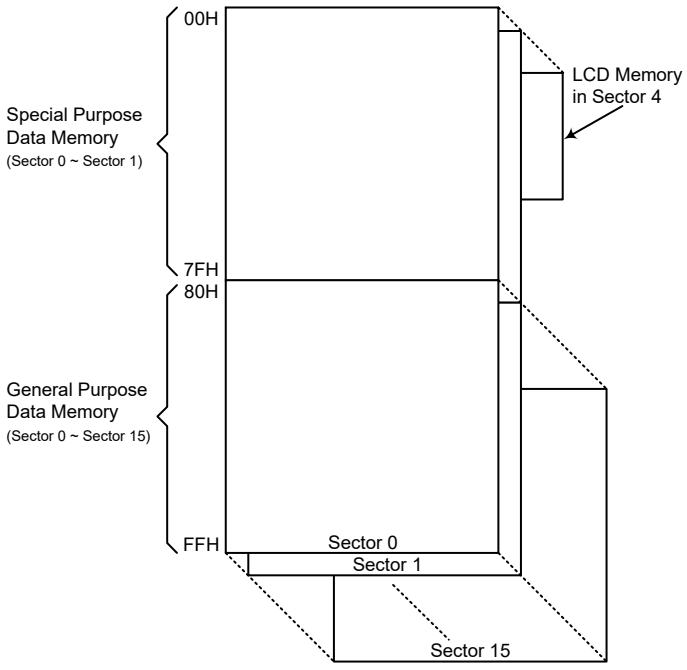
The device also provides a dedicated memory area for the LCD display data storage. In this chapter, only the General Purpose Data Memory and the Special Function Register Data Memory are introduced. Details about the LCD Data Memory can be obtained in the LCD Driver section.

Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide Memory. Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH.

Special Purpose Data Memory	General Purpose Data Memory		LCD Data Memory
Located Sectors	Capacity	Address	Address
Sector 0: 00H~7FH Sector 1: 00H~7FH	2048×8	Sector 0: 80H~FFH Sector 1: 80H~FFH : Sector 15: 80H~FFH	Sector 4: 00H~23H

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. 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 instructions which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address “m” in the extended instructions has 12 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
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH	PBP	
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	SCC	
11H	HIRCC	
12H	HXTC	
13H	LXTC	
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	RSTC	
19H	LVRC	
1AH		
1BH	MFI0	
1CH	MFI1	
1DH	MFI2	
1EH	WDTC	
1FH	INTEG	
20H	INTC0	
21H	INTC1	
22H	INTC2	
23H	INTC3	
24H	PB	
25H	PBC	
26H	PBPU	
27H	PC	
28H	PCC	
29H	PCPU	
2AH		
2BH		
2CH	PSCR	
2DH	TB0C	
2EH	TB1C	
2FH		
30H	SIM0C0	PTM0C0
31H	SIM0C1/U0UCR1	PTM0C1
32H	SIM0C2/SIM0A/U0UCR2	PTM0DL
33H	SIM0D/U0TXR_RXR	PTM0DH
34H	SIM0TOC/U0BRG	PTM0AL
35H	U0USR	PTM0AH
36H	U0UCR3	PTM0RPL
37H	SIM1C0	PTM0RPH
38H	SIM1C1/U1UCR1	PTM1C0
39H	SIM1C2/SIM1A/U1UCR2	PTM1C1
3AH	SIM1D/U1TXR_RXR	PTM1DL
3BH	SIM1TOC/U1BRG	PTM1DH
3CH	U1USR	PTM1AL
3DH	U1UCR3	PTM1AH
3EH	SPIC0	PTM1RPL
3FH	SPIC1	PTM1RPH

	Sector 0	Sector 1
40H	SPID	EEC
41H	EEAL	
42H	EEAH	
43H	EED	FC0
44H		FC1
45H		FC2
46H		FARL
47H		FARH
48H		FD0L
49H	STMC0	FD0H
4AH	STMC1	FD1L
4BH	STMDL	FD1H
4CH	STMDH	FD2L
4DH	STMAL	FD2H
4EH	STMAH	FD3L
4FH	STMRP	FD3H
50H	ATMC0	
51H	ATMC1	
52H	ATMDL	
53H	ATMDH	PAS0
54H	ATMAL	PAS1
55H	ATMAH	PBS0
56H	ATMBL	PBS1
57H	ATMBH	PCS0
58H	ATMRP	PCS1
59H		PDS0
5AH		PDS1
5BH		PES0
5CH		PES1
5DH		PFS0
5EH		PFS1
5FH		PGS0
60H	LCDC0	PGS1
61H	LCDC2	PHS0
62H		PHS1
63H		COMS
64H	SADOL	PMPS
65H	SADOH	
66H	SADC0	
67H	SADC1	
68H	IREFC	
69H	PVREF	
6AH	OPA1C	
6BH	OPA2C	
6CH	GSC	
6DH	AFEDAC	
6EH	AFEDAL	
6FH	AFEDAH	
70H	PD	
71H	PDC	
72H	PDPU	
73H	PE	
74H	PEC	
75H	PEPU	
76H	PF	
77H	PFC	
78H	PFPU	
79H	PG	
7AH	PGC	
7BH	PGPU	
7CH	PH	
7DH	PHC	
7EH	PHPU	
7FH	ORMC	

: Unused, read as 00H

: Reserved, cannot be changed

Special Purpose Data Memory Structure

Special Function Register Description

Most of the Special Function Register details will be described in the relevant functional section. 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 MP1L/MP1H register pair and IAR2 register together with 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 indirectly will return a result of "00H" and writing to the registers indirectly will result in no operation.

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

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 microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Sector 0, while MP1L/MP1H together with IAR1 and MP2L/MP2H together with IAR2 are used to access data from all 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.

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 several banks, Bank 0~Bank 3. 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	—	—	—	—	—	—	PBP1	PBP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PBP1~PBP0**: Program Memory Bank Selection
00: Bank 0
01: Bank 1
10: Bank 2
11: Bank 3

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 microcontroller.

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

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

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- PDF is cleared by a system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction.
- TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.
- 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 content of the status register is 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 Z 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 2048×8 bits for this device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using a pair of address registers and a data register and a single control register which is located in sector 1.

EEPROM Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEAL	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH	—	—	—	—	—	EEAH2	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 bit 7 ~ bit 0

- **EEAH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	EEAH2	EEAH1	EEAH0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **EEAH2~EEAH0**: Data EEPROM address high byte bit 2 ~ bit 0

• **EED Register**

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

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

• **EEC Register**

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

Bit 7 **EWERTS**: Data EEPROM Erase time and Write time selection
0: Erase time is 3.2ms / Page write time is 2.2ms / byte write time is 5.4ms
1: Erase time is 3.7ms / Page write time is 3.0ms / byte write time is 6.7ms

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 bit 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.
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 only the RD bit high will not initiate a read operation if the RDEN bit is not set high. When the read cycle terminates, the EEPROM data can be read from the EED register and the RD bit will automatically be cleared to zero. The data will remain in the EED register until another read or write operation is executed. The application program can poll the RD bit to determine when the data is valid for reading.

Page Read Mode

The EEPROM page read operation can be executed when the mode selection bit, MODE, is set high. The page size can be up to 16 bytes for the page read operation. For a page read operation the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM page read operation. Note that setting only the RD bit high will not initiate a read operation if the RDEN bit is not set high. When the current byte read cycle terminates, the EEPROM data can be read from the EED register and then the current address will be incremented by one automatically. After this the RD bit will automatically be cleared to zero. The data which is stored in the next EEPROM address can continuously be read out when the RD bit is set high again 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 7 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 7-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 7 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 7-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not "roll over".

For page erase operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers and then write a dummy data into the EED register to tag address. After a dummy data is written, the current address is tagged and the lower 4-bit address value will be automatically incremented by one. As the maximum data length for a page is 16 bytes, when the address reaches the page boundary, 0FH, the address will not be further incremented but stop at the last address of the page. Note that the dummy write operations must be implemented to determine which addresses to be erased. When all desired addresses are tagged, then the EREN bit in the EEC register can be set high to enable erase operations and the ER bit must be immediately set high to initiate the EEPROM erase process. These two instructions must be executed in two consecutive instruction cycles to activate an erase operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing an erase operation and then set again after a valid erase activation procedure has completed.

As the EEPROM erase cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been erased from the EEPROM. Detecting when the erase cycle has finished can be implemented either by polling the ER bit in the EEC register or by using the EEPROM interrupt. When the erase cycle terminates, the ER bit will be automatically cleared to zero by the microcontroller, indicating 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 and the data to be written should be placed in the EED register. To write data to the EEPROM, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, indicating 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 7 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 7-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”. At this point any data write operations to the EED register will be invalid.

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

operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, indicating 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 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 both be set. If the global, EEPROM and Multi-function interrupts are enabled and the stack is not full, a jump to the associated Multi-function Interrupt vector will take place. When the interrupt is serviced only the Multi-function interrupt flag will be automatically reset, the EEPROM interrupt flag must be manually reset by the application program. More details can be obtained in the Interrupt section.

Programming Considerations

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

When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write 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 /write/erase operation is totally complete. Otherwise, the EEPROM read /write/erase operation will fail. During an erase or write operation, the IAP function should not be used, which may lead to a false modification to the data in the EEPROM and Program memory.

Programming Examples

Reading a Data Byte from the EEPROM – polling method

```
MOV A, 040H              ; setup memory pointer low byte MP1L
MOV MP1L, A              ; MP1L points to EEC register
MOV A, 01H               ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4               ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H    ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L    ; user defined low byte address
MOV EEAL, A
SET IAR1.1               ; set RDEN bit, enable read operations
SET IAR1.0               ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0               ; check for read cycle end
JMP BACK
CLR IAR1                 ; disable EEPROM read function
CLR MP1H
MOV A, EED               ; move read data to register
MOV READ_DATA, A
```

Reading a Data Page from the EEPROM – polling method

```
MOV A, 040H              ; setup memory pointer low byte MP1L
MOV MP1L, A              ; 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, 040H              ; 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, 040H              ; 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, 040H              ; 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 types offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving. Oscillator selections and operation are selected through a combination of configuration options and relevant control registers.

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency 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.

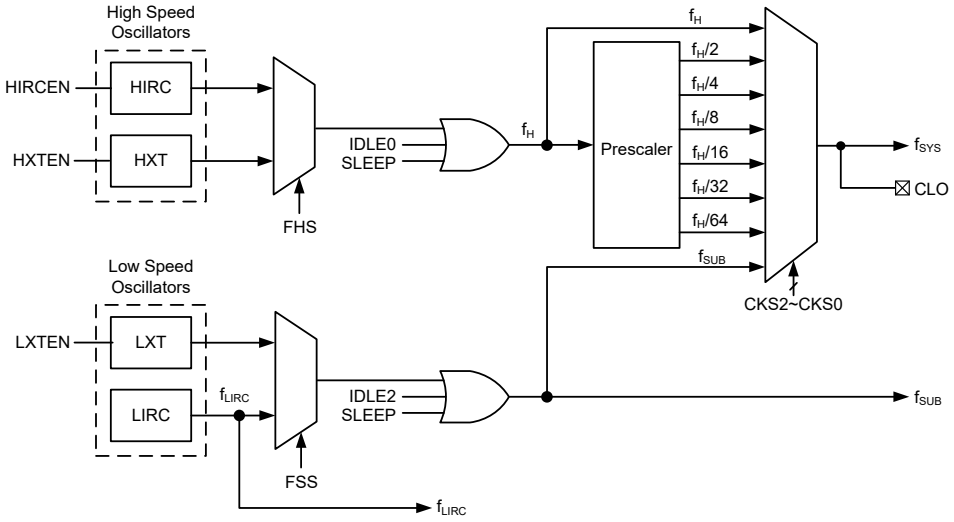
Type	Name	Frequency	Pins
External Crystal/Resonator Oscillator	HXT	400kHz~16MHz	OSC1/OSC2
Internal High Speed RC Oscillator	HIRC	4/8/12MHz	—
External Low Speed Crystal Oscillator	LXT	32.768kHz	XT1/XT2
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

System Clock Configurations

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

The actual source clock used for the low speed oscillators is chosen via the FSS bit in the SCC register while for the high speed oscillators the source clock is selected by the FHS bit in the SCC register. The frequency of the slow speed or high speed system clock is determined using the CKS2~CKS0 bits in the SCC register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators.

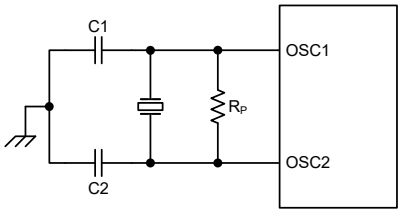


System Clock Configurations

External Crystal/Resonator Oscillator – HXT

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

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



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

Crystal/Resonator Oscillator – HXT

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

Crystal Recommended Capacitor Values

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 4MHz, 8MHz and 12MHz which are selected using a configuration option. The HIRC1~HIRC0 bits in the HIRCC register must also be setup to match the selected configuration option frequency. Setting up these bits is necessary to ensure that the HIRC frequency accuracy specified in the A.C. Characteristics is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that this internal system clock requires no external pins for its operation.

External 32.768kHz Crystal Oscillator – LXT

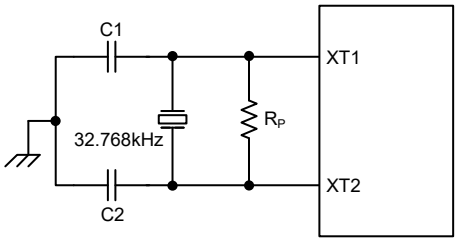
The External 32.768kHz Crystal System Oscillator is one of the low frequency oscillator choices. This clock source has a fixed frequency of 32.768kHz and requires a 32.768kHz crystal to be connected between pins XT1 and XT2. The external resistor and capacitor components connected to the 32.768kHz crystal 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.

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 two modes, the Speed-Up Mode and the Low-Power Mode. The mode selection is executed using the LXTSP bit in the LXTC register

LXTSP Bit	LXT Operating Mode
0	Low Power
1	Speed Up

When the LXTSP bit is set high, the LXT Speed Up Mode will be enabled. In the Speed-Up Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up, it can be placed into the Low-Power Mode by clearing the LXTSP bit to zero and the oscillator will continue to run but with reduced current consumption. It is important to note that the LXT operating mode switching must be properly controlled before the LXT oscillator clock is selected as the system clock source. Once the LXT oscillator clock is selected as the system clock source using the CKS bit field and FSS bit in the SCC register, the LXT oscillator operating mode cannot be changed.

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

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is one of the low frequency oscillator choices. 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.

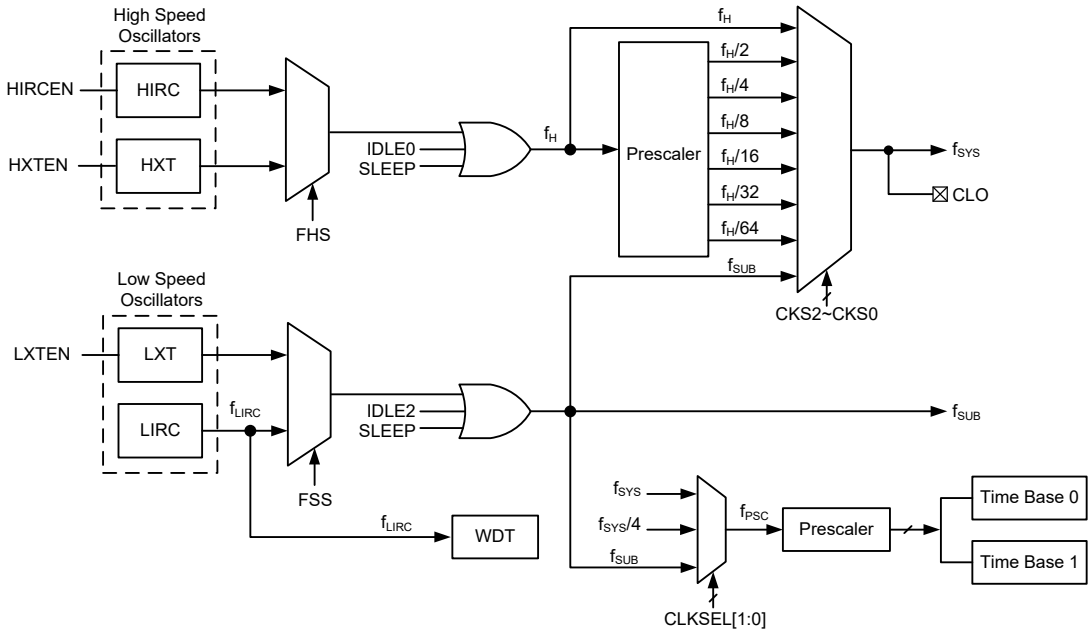
Operating Modes and System Clocks

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

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



Device Clock Configurations

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

System Operation Modes

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

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

"x": don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.
2. In the SLEEP mode, the f_{LIRC} clock can be on or off which is controlled by the WDT function being enabled or disabled.

FAST Mode

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

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from either the LIRC or LXT oscillator 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 bits in the SCC register are both low. In the SLEEP mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral function will also be stopped. However the f_{LIRC} clock will continue to operate if the 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 on to drive some peripheral functions.

IDLE1 Mode

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

Control Registers

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

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

System Operating Mode Control Register List

• SCC Register

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

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

Bit 4 Unimplemented, read as "0"

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

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

Bit 1 **FHIDEN**: High Frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the selected 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.

• **HIRCC Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection
00: 4MHz
01: 8MHz
10: 12MHz
11: 4MHz
When the HIRC oscillator is enabled or the HIRC frequency selection is changed by the application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1.
It is recommended that the HIRC frequency selected by these two bits should be the same with the frequency determined by the configuration option to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Bit 1 **HIRCF**: HIRC oscillator stable flag
0: Unstable
1: 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

• **HXTC Register**

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

Bit 7~3 Unimplemented, read as "0"

Bit 2 **HXTM**: HXT mode selection
0: HXT frequency ≤ 10MHz
1: HXT frequency >10MHz
Note that this bit should be setup correctly according to the used HXT frequency, otherwise, the oscillation perfermance will be not well.
The HXTM bit should be properly configured before the HXT function is enabled. When the pin-shared functions have been configured to select the OSC1/OSC2 pin functions and the HXTEN bit has been set to 1 to enable the HXT oscillator, then it is invalid to change the value of the HXTM bit. When the OSC1 or OSC2 pin function is disabled, then the HXTM bit can be changed by software, regardless of the HXTEN bit value.

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

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

• **LXTC Register**

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

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LXTSP**: LXT Speed up control
0: Disable – Low power
1: Enable – Speed up
This bit is used to control whether the LXT oscillator is operating in the low power or Speed-Up mode. When the LXTSP bit is set high, the LXT oscillator will oscillate quickly but consume more power. If the LXTSP bit is cleared to zero, the LXT oscillator will consume less power but take longer time to stablise. It is important to note that this bit cannot be changed after the LXT oscillator is selected as the system clock source using the CKS2~CKS0 and FSS bits in the SCC register.

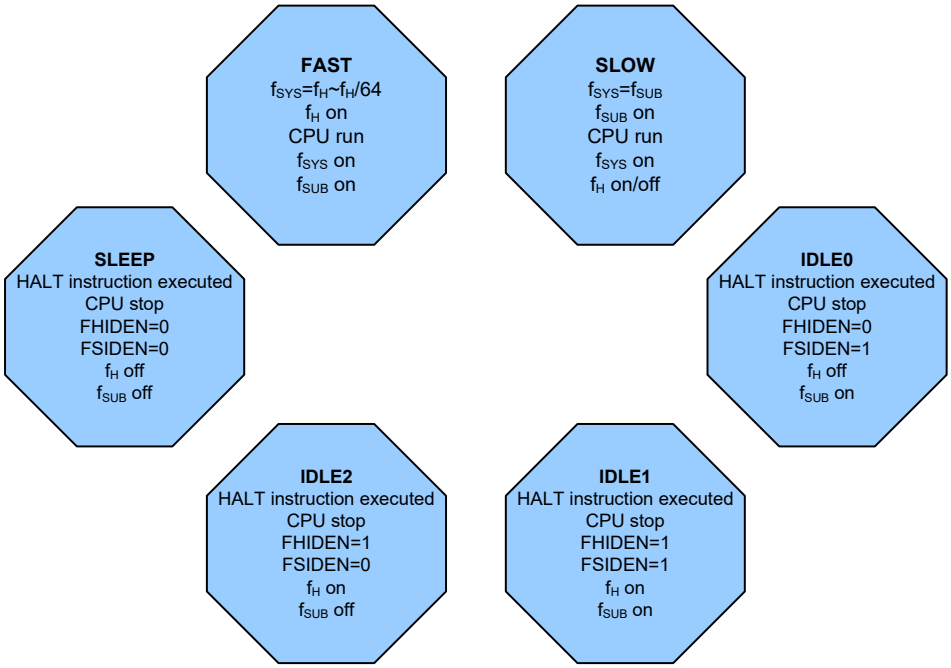
Bit 1 **LXTF**: LXT oscillator stable flag
0: Unstable
1: Stable
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set to 1 to enable the LXT oscillator, the LXTF bit will first be cleared to 0 and then set to 1 after the LXT oscillator is stable.

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

Operating Mode Switching

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

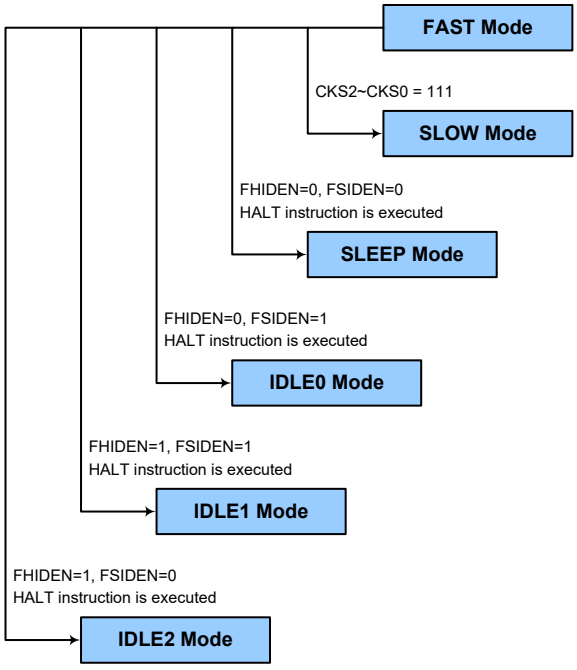
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



FAST Mode to SLOW Mode Switching

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

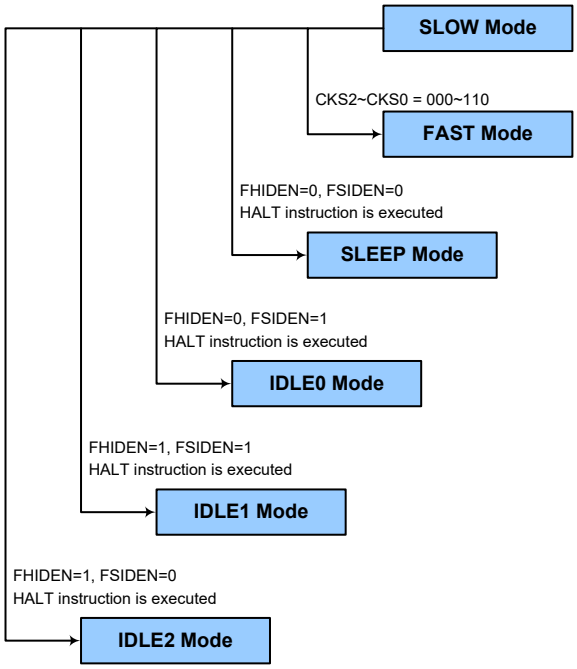
The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires these oscillators to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

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

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



Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "0". 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 LXT or 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 zero if the device experiences a system power-up or executes the clear Watchdog Timer instruction. If the device is woken up by a WDT overflow, a Watchdog Timer reset will be initiated and the TO flag will be set to 1. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

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

Programming Considerations

The start-up time of the high speed and low speed oscillators are different. For example, if the system is woken up from the SLEEP Mode and both the HXT and LXT oscillators need to start-up from an off state. If the device is woken up from the SLEEP Mode to the FAST Mode, the high speed system oscillator needs an SST period. The device will execute first instruction after HIRCF/HXTF is "1". At this time, the LXT oscillator may not be stability if f_{SUB} is from the LXT oscillator. The same situation occurs in the power-on state. The LXT oscillator is not ready yet when the first instruction is executed.

There are peripheral functions, such as TMs, for which the f_{SYS} is used. If the system clock source is switched from f_H to f_{SUB}, the clock source to the peripheral functions mentioned above will change accordingly.

Watchdog Timer

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

Watchdog Timer Clock Source

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

Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period as well as the 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 enable control
10101: Disable
01010: Enable
Other values: Generates an MCU reset
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: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$
These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period.

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Refer to the Internal Reset Control section.

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

Bit 1 **LRF**: LVRC register software reset flag
Refer to the Low Voltage Reset section.

Bit 0 **WRF**: WDTC register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the WDTC control register software reset and cleared by the application program. This bit can only be cleared to zero by 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 enable/disable control and reset control of the Watchdog Timer. 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.

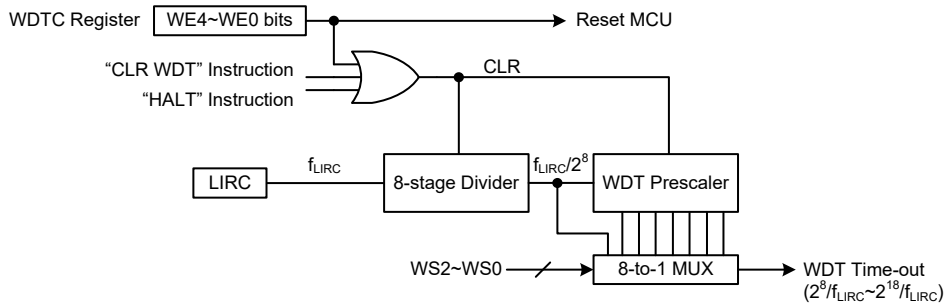
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

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 WDTC register 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.

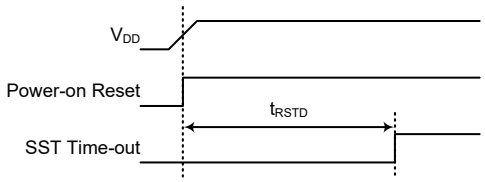
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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



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

Power-On Reset Timing Chart

Internal Reset Control

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

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

Internal Reset Function Control

• RSTC Register

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

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

• RSTFC Register

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the RSTC control register software reset and cleared by the application program. This bit can only be cleared to zero by application program.

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

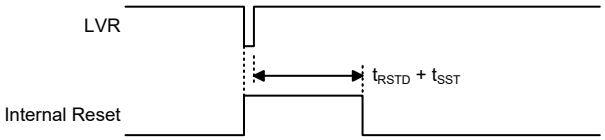
Bit 1 **LRF**: LVRC register software reset flag
Refer to the Low Voltage Reset section.

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

Low Voltage Reset – LVR

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

The LVR function is always enabled except in the SLEEP and IDLE mode with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits value is modified to any other value, 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. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.



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

Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Other values: Generates an MCU reset

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

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

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Refer to the Internal Reset Control section.

Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set to 1 when a specific low voltage reset condition occurs. This bit can only be cleared to zero by application program.

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

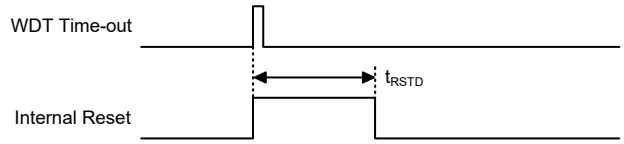
Bit 0 **WRF**: WDTC register software reset flag
Refer to the Watchdog Timer Control Register section.

IAP Reset

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

Watchdog Time-out Reset during Normal Operation

After a Watchdog time-out Reset during normal operation, the Watchdog time-out flag TO will be set to "1".

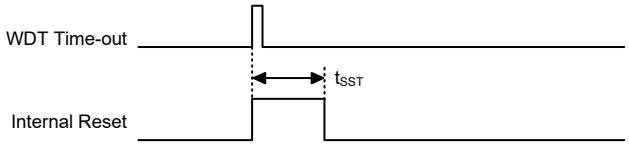


Note: t_{RSTD} is power-on delay specified in System Start Up Time Characteristics.

WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during SLEEP or IDLE 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	Clear after reset, WDT begins counting
Timer Modules	Timer Modules will be turned off
Input/Output Ports	I/O ports will be setup as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

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

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	-xxx xxxx	-uuu uuuu	-uuu uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
PBP	---- --00	---- --00	---- --uu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- 0x00	---- uuuu	---- uuuu
SCC	000- 0000	000- 0000	uuu- uuuu
HIRCC	---- 0001	---- 0001	---- uuuu
HXTC	---- -000	---- -000	---- -uuu
LXTC	---- -000	---- -000	---- -uuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PAWU	0000 0000	0000 0000	uuuu uuuu
RSTC	0101 0101	0101 0101	uuuu uuuu
LVRC	0101 0101	0101 0101	uuuu uuuu
MFI0	0000 0000	0000 0000	uuuu uuuu
MFI1	-000 -000	-000 -000	-uuu -uuu
MFI2	-000 -000	-000 -000	-uuu -uuu
WDTC	0101 0011	0101 0011	uuuu uuuu
INTEG	0000 0000	0000 0000	uuuu uuuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	--00 --00	--00 --00	--uu --uu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
PC	-111 1111	-111 1111	-uuu uuuu
PCC	-111 1111	-111 1111	-uuu uuuu
PCPU	-000 0000	-000 0000	-uuu uuuu
PSCR	---- --00	---- --00	---- --uu
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
SIM0C0	1110 0000	1110 0000	uuuu uuuu
SIM0C1 (U0MD=0)	1000 0001	1000 0001	uuuu uuuu
U0UCR1* (U0MD=1)	0000 00x0	0000 00x0	uuuu uuuu
SIM0C2/SIM0A/U0UCR2	0000 0000	0000 0000	uuuu uuuu
SIM0D/U0TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIM0TOC (U0MD=0)	0000 0000	0000 0000	uuuu uuuu
U0BRG* (U0MD=1)	xxxx xxxx	xxxx xxxx	uuuu uuuu
U0USR	0000 1011	0000 1011	uuuu uuuu
U0UCR3	---- ---0	---- ---0	---- ---u
SIM1C0	1110 0000	1110 0000	uuuu uuuu
SIM1C1 (U1MD=0)	1000 0001	1000 0001	uuuu uuuu
U1UCR1* (U1MD=1)	0000 00x0	0000 00x0	uuuu uuuu
SIM1C2/SIM1A/U1UCR2	0000 0000	0000 0000	uuuu uuuu
SIM1D/U1TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIM1TOC (U1MD=0)	0000 0000	0000 0000	uuuu uuuu
U1BRG* (U1MD=1)	xxxx xxxx	xxxx xxxx	uuuu uuuu
U1USR	0000 1011	0000 1011	uuuu uuuu
U1UCR3	---- ---0	---- ---0	---- ---u
SPIC0	111- --00	111- --00	uuu- --uu
SPIC1	--00 0000	--00 0000	--uu uuuu
SPID	xxxx xxxx	xxxx xxxx	uuuu uuuu
EEAL	0000 0000	0000 0000	uuuu uuuu
EEAH	---- -000	---- -000	---- -uuu
EED	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

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
STMAH	0000 0000	0000 0000	uuuu uuuu
STMRP	0000 0000	0000 0000	uuuu uuuu
ATMC0	0000 0--0	0000 0--0	uuuu u--u
ATMC1	0000 0000	0000 0000	uuuu uuuu
ATMDL	0000 0000	0000 0000	uuuu uuuu
ATMDH	---- --00	---- --00	---- --uu
ATMAL	0000 0000	0000 0000	uuuu uuuu
ATMAH	---- --00	---- --00	---- --uu
ATMBL	0000 0000	0000 0000	uuuu uuuu
ATMBH	---- --00	---- --00	---- --uu
ATMRP	0000 0000	0000 0000	uuuu uuuu
LCDC0	0-00 ---0	0-00 ---0	u-uu ---u
LCDC2	000- -00-	000- -00-	uuu- -uu-
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	uuuu uuuu
IREFC	00-0 0-00	00-0 0-00	uu-u u-uu
PVREF	0000 0000	0000 0000	uuuu uuuu
OPA1C	0000 0000	0000 0000	uuuu uuuu
OPA2C	0000 0000	0000 0000	uuuu uuuu
GSC	---- 0000	---- 0000	---- uuuu
AFEDAC	---- --00	---- --00	---- --uu
AFEDAL	0000 ----	0000 ----	uuuu ----
AFEDAH	0000 0000	0000 0000	uuuu uuuu
PD	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	uuuu uuuu
PDPU	0000 0000	0000 0000	uuuu uuuu
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	--11 1111	--11 1111	--uu uuuu
PGC	--11 1111	--11 1111	--uu uuuu
PGPU	--00 0000	--00 0000	--uu uuuu
PH	-111 11--	-111 11--	-uuu uu--
PHC	-111 11--	-111 11--	-uuu uu--
ORMC	0000 0000	0000 0000	0000 0000
PHPU	-000 00--	-000 00--	-uuu uu--
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

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PTM0AL	0000 0000	0000 0000	uuuu uuuu
PTM0AH	---- --00	---- --00	---- --uu
PTM0RPL	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	---- --00	---- --00	---- --uu
PTM1C0	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	uuuu uuuu
PTM1DL	0000 0000	0000 0000	uuuu uuuu
PTM1DH	---- --00	---- --00	---- --uu
PTM1AL	0000 0000	0000 0000	uuuu uuuu
PTM1AH	---- --00	---- --00	---- --uu
PTM1RPL	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	---- --00	---- --00	---- --uu
EEC	0000 0000	0000 0000	uuuu uuuu
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	-000 0000	-000 0000	-uuu uuuu
FD0L	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	uuuu uuuu
PAS0	00--00--	00--00--	uu-- uu--
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 0000	0000 0000	uuuu uuuu
PBS1	0000 0000	0000 0000	uuuu uuuu
PCS0	0000 0000	0000 0000	uuuu uuuu
PCS1	--00 0000	--00 0000	--uu uuuu
PDS0	0000 0000	0000 0000	uuuu uuuu
PDS1	0000 0000	0000 0000	uuuu uuuu
PES0	0000 0000	0000 0000	uuuu uuuu
PES1	0000 0000	0000 0000	uuuu uuuu
PFS0	0000 0000	0000 0000	uuuu uuuu
PFS1	0000 0000	0000 0000	uuuu uuuu
PGS0	0000 0000	0000 0000	uuuu uuuu
PGS1	---- 0000	---- 0000	---- uuuu
PHS0	0000 ----	0000 ----	uuuu ----
PHS1	--00 0000	--00 0000	--uu uuuu
COMS	---- --00	---- --00	---- --uu
PMPS	---- --00	---- --00	---- --uu

Note: “u” stands for unchanged
“x” stands for unknown
“-” stands for unimplemented
“*”: The UnUCR1 and SIMnC1 registers share the same memory address while the UnBRG and SIMnTOC registers share the same memory address. The default value of the UnUCR1 or UnBRG register can be obtained when the UnMD bit is set high by application program after a reset.

Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	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	—	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	—	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	—	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PF	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0
PG	—	—	PG5	PG4	PG3	PG2	PG1	PG0
PGC	—	—	PGC5	PGC4	PGC3	PGC2	PGC1	PGC0
PGPU	—	—	PGPU5	PGPU4	PGPU3	PGPU2	PGPU1	PGPU0
PH	—	PH6	PH5	PH4	PH3	PH2	—	—
PHC	—	PHC6	PHC5	PHC4	PHC3	PHC2	—	—
PHPU	—	PHPU6	PHPU5	PHPU4	PHPU3	PHPU2	—	—

"—": 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 relevant pull-high control registers and are implemented using weak PMOS transistors.

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

- **PxPU Register**

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

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

The PxPUn bit is used to control the pin pull-high function. Here the "x" 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.

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 pin Wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

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

• **PxC Register**

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

This device supports different I/O port power source selections for PD0~PD3. The port power can come from either the power pin VDD or VDDIO, which is determined using the PMPS1~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. With the exception of $\overline{RES}$/OCDS, the multi-power function is only effective when the pin is set to have a digital input or output function.

• **PMPS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PMPS1	PMPS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PMPS1~PMPS0**: PD3~PD0 pin power supply selection
00/01: V_{DD}
10/11: V_{DDIO}
If the PB6 pin is switched to the VDDIO function, and the PMPS1 and PMPS0 bits are set to "10" or "11", the VDDIO pin input voltage can be used for PD3~PD0 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. Each device includes Port "x" pin-shared function selection register "n", labeled as PxSn, which can select the desired functions of the multi-function pin-shared pins. In addition, the device also contains a COMS register which is used to select the actual pin function when the LCD SEG and COM functions share the same pin.

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, 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	PAS07	PAS06	—	—	PAS03	PAS02	—	—
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	—	—	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
PDS1	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
PES0	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
PES1	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
PFS0	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
PFS1	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10
PGS0	PGS07	PGS06	PGS05	PGS04	PGS03	PGS02	PGS01	PGS00
PGS1	—	—	—	—	PGS13	PGS12	PGS11	PGS10
PHS0	PHS07	PHS06	PHS05	PHS04	—	—	—	—
PHS1	—	—	PHS15	PHS14	PHS13	PHS12	PHS11	PHS10
COMS	—	—	—	—	—	—	COMS1	COMS0

Pin-shared Function Selection Register List

- **PAS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	—	—	PAS03	PAS02	—	—
R/W	R/W	R/W	—	—	R/W	R/W	—	—
POR	0	0	—	—	0	0	—	—

Bit 7~6 **PAS07~PAS06**: PA3 pin-shared function selection
00: PA3
01: SCK0/SCL0
10: PA3
11: PA3

Bit 5~4 Unimplemented, read as "0"

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00: PA1
01: $\overline{SCS0}$
10: PA1
11: PA1

Bit 1~0 Unimplemented, read as "0"

• **PAS1 Register**

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

Bit 7~6 **PAS17~PAS16**: PA7 pin-shared function selection
00: PA7/INT1
01: OP2N
10: PA7/INT1
11: PA7/INT1

Bit 5~4 **PAS15~PAS14**: PA6 pin-shared function selection
00: PA6/INT0
01: VG
10: PA6/INT0
11: PA6/INT0

Bit 3~2 **PAS13~PAS12**: PA5 pin-shared function selection
00: PA5
01: PA5
10: PA5
11: ATP/ATP_PWM1

Bit 1~0 **PAS11~PAS10**: PA4 pin-shared function selection
00: PA4
01: PA4
10: PA4
11: ATPB/ATP_PWM2

• **PBS0 Register**

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

Bit 7~6 **PBS07~PBS06**: PB3 pin-shared function selection
00: PB3
01: PB3
10: PB3
11: AN1

Bit 5~4 **PBS05~PBS04**: PB2 pin-shared function selection
00: PB2
01: PB2
10: PB2
11: AN0

Bit 3~2 **PBS03~PBS02**: PB1 pin-shared function selection
00: PB1
01: PB1
10: PB1
11: OP1N

Bit 1~0 **PBS01~PBS00**: PB0 pin-shared function selection
00: PB0
01: PB0
10: PB0
11: OP2O

• **PBS1 Register**

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

Bit 7~6 **PBS17~PBS16**: PB7 pin-shared function selection
00: PB7
01: SCK1/SCL1
10: PB7
11: AN5

Bit 5~4 **PBS15~PBS14**: PB6 pin-shared function selection
00: PB6
01: VDDIO
10: VREF
11: AN4

Bit 3~2 **PBS13~PBS12**: PB5 pin-shared function selection
00: PB5
01: PTP0B
10: PB5
11: AN3

Bit 1~0 **PBS11~PBS10**: PB4 pin-shared function selection
00: PB4
01: PTP0
10: PB4
11: AN2

• **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: XT1
10: PC3
11: PC3

Bit 5~4 **PCS05~PCS04**: PC2 pin-shared function selection
00: PC2/INT2
01: SDO1/UTX1
10: PC2/INT2
11: PC2/INT2

Bit 3~2 **PCS03~PCS02**: PC1 pin-shared function selection
00: PC1/INT3
01: SDI1/SDA1/URX1/UTX1
10: PC1/INT3
11: PC1/INT3

Bit 1~0 **PCS01~PCS00**: PC0 pin-shared function selection
00: PC0
01: SDI0/SDA0/URX0/UTX0
10: PC0
11: PC0

• **PCS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
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 **PCS15~PCS14**: PC6 pin-shared function selection
00: PC6
01: SDO0/UTX0
10: PC6
11: PC6

Bit 3~2 **PCS13~PCS12**: PC5 pin-shared function selection
00: PC5/ATCK
01: $\overline{\text{SCS1}}$
10: PC5/ATCK
11: PC5/ATCK

Bit 1~0 **PCS11~PCS10**: PC4 pin-shared function selection
00: PC4
01: XT2
10: PC4
11: PC4

• **PDS0 Register**

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

Bit 7~6 **PDS07~PDS06**: PD3 pin-shared function selection
00: PD3
01: PD3
10: $\overline{\text{SPISCS}}$
11: SEG3

Bit 5~4 **PDS05~PDS04**: PD2 pin-shared function selection
00: PD2/PTCK0
01: PD2/PTCK0
10: SPISDO
11: SEG2

Bit 3~2 **PDS03~PDS02**: PD1 pin-shared function selection
00: PD1
01: SPISDI
10: PTP0B
11: SEG1

Bit 1~0 **PDS01~PDS00**: PD0 pin-shared function selection
00: PD0
01: SPISCK
10: PTP0
11: SEG0

• **PDS1 Register**

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

Bit 7~6 **PDS17~PDS16**: PD7 pin-shared function selection
00: PD7
01: OSC2
10: PD7
11: SEG7

Bit 5~4 **PDS15~PDS14**: PD6 pin-shared function selection
00: PD6
01: OSC1
10: PD6
11: SEG6

Bit 3~2 **PDS13~PDS12**: PD5 pin-shared function selection
00: PD5
01: PD5
10: ATPB/ATP_PWM2
11: SEG5

Bit 1~0 **PDS11~PDS10**: PD4 pin-shared function selection
00: PD4
01: PD4
10: ATP/ATP_PWM1
11: SEG4

• **PES0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PES07~PES06**: PE3 pin-shared function selection
00: PE3
01: PE3
10: PTP1
11: SEG11

Bit 5~4 **PES05~PES04**: PE2 pin-shared function selection
00: PE2/STCK
01: PE2/STCK
10: PE2/STCK
11: SEG10

Bit 3~2 **PES03~PES02**: PE1 pin-shared function selection
00: PE1
01: PE1
10: STPB
11: SEG9

Bit 1~0 **PES01~PES00**: PE0 pin-shared function selection
00: PE0
01: PE0
10: STP
11: SEG8

• **PES1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PES17~PES16**: PE7 pin-shared function selection
00: PE7
01: PE7
10: PE7
11: SEG15

Bit 5~4 **PES15~PES14**: PE6 pin-shared function selection
00: PE6
01: PE6
10: PE6
11: SEG14

Bit 3~2 **PES13~PES12**: PE5 pin-shared function selection
00: PE5/PTCK1
01: PE5/PTCK1
10: PE5/PTCK1
11: SEG13

Bit 1~0 **PES11~PES10**: PE4 pin-shared function selection
00: PE4
01: PE4
10: PTP1B
11: SEG12

• **PFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PFS07~PFS06**: PF3 pin-shared function selection
00: PF3
01: PF3
10: PF3
11: SEG19

Bit 5~4 **PFS05~PFS04**: PF2 pin-shared function selection
00: PF2
01: PF2
10: PF2
11: SEG18

Bit 3~2 **PFS03~PFS02**: PF1 pin-shared function selection
00: PF1
01: PF1
10: PF1
11: SEG17

Bit 1~0 **PFS01~PFS00**: PF0 pin-shared function selection
00: PF0
01: PF0
10: PF0
11: SEG16

• **PFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PFS17~PFS16**: PF7 pin-shared function selection
00: PF7
01: PF7
10: PF7
11: SEG23

Bit 5~4 **PFS15~PFS14**: PF6 pin-shared function selection
00: PF6
01: PF6
10: PF6
11: SEG22

Bit 3~2 **PFS13~PFS12**: PF5 pin-shared function selection
00: PF5
01: PF5
10: PF5
11: SEG21

Bit 1~0 **PFS11~PFS10**: PF4 pin-shared function selection
00: PF4
01: PF4
10: PF4
11: SEG20

• **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: PG3
10: PG3
11: SEG27

Bit 5~4 **PGS05~PGS04**: PG2 pin-shared function selection
00: PG2
01: PG2
10: PG2
11: SEG26

Bit 3~2 **PGS03~PGS02**: PG1 pin-shared function selection
00: PG1
01: PG1
10: PG1
11: SEG25

Bit 1~0 **PGS01~PGS00**: PG0 pin-shared function selection
00: PG0
01: PG0
10: PG0
11: SEG24

• **PGS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PGS13	PGS12	PGS11	PGS10
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 **PGS13~PGS12**: PG5 pin-shared function selection
00: PG5
01: PG5
10: COM6
11: SEG29

Bit 1~0 **PGS11~PGS10**: PG4 pin-shared function selection
00: PG4
01: PG4
10: COM7
11: SEG28

• **PHS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PHS07	PHS06	PHS05	PHS04	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

Bit 7~6 **PHS07~PHS06**: PH3 pin-shared function selection
00: PH3
01: PH3
10: PH3
11: SEG32

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

Bit 3~0 Unimplemented, read as "0"

• **PHS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PHS15	PHS14	PHS13	PHS12	PHS11	PHS10
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 **PHS15~PHS14**: PH6 pin-shared function selection
00: PH6
01: PH6
10: PH6
11: SEG35

Bit 3~2 **PHS13~PHS12**: PH5 pin-shared function selection
00: PH5
01: PH5
10: PH5
11: SEG34

Bit 1~0 **PHS11~PHS10**: PH4 pin-shared function selection
00: PH4
01: PH4
10: PH4
11: SEG33

• **COMS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	COMS1	COMS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

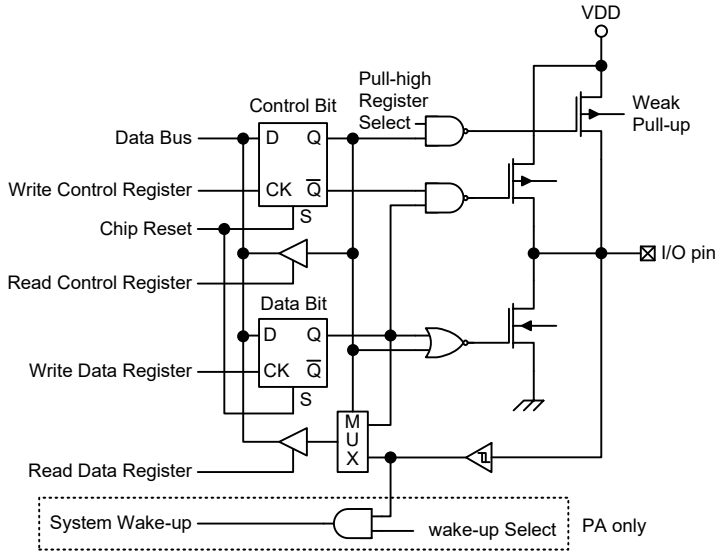
Bit 7~2 Unimplemented, read as "0"

Bit 1 **COMS1**: COM5/SEG30 pin-shared function selection
0: SEG30
1: COM5

Bit 0 **COMS0**: COM4/SEG31 pin-shared function selection
0: SEG31
1: COM4

I/O Pin Structures

The accompanying diagram illustrates the internal structures of the I/O logic function. As the exact logical construction of the I/O pin will differ from this drawing, it is supplied as a guide only to assist with the functional understanding of the logic function I/O pins. The wide range of pin-shared structures does not permit all types to be shown.



Logic Function Input/Output Structure

Programming Considerations

Within the user program, one of the things first to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set to high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port

data registers are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

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

Timer Modules – TM

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

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

Introduction

The device contains four TMs and each individual TM is categorised as a certain type, namely Standard Type TM, Periodic Type TM or Audio Type TM. The number of the Audio type and Standard type TM is one, having a reference name of STM and ATM respectively. The remaining two TMs are Periodic type with a reference name of PTM0 and PTM1. Although similar in nature, the different TM types vary in their feature complexity. The common features to all of the Standard, Periodic and Audio Type TMs will be described in this section and the detailed operation regarding each of the TM types will be described in separate sections. The main features and differences between the three types of TMs are summarised in the accompanying table.

TM Function	STM	PTM	ATM
Timer/Counter	√	√	√
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 S or P or A type TM and "n" stands for the specific TM serial number. For STM and ATM there is no serial number "n" in the relevant pin or control register bit names since there are only an STM and an ATM 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 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. As the Audio Type TM has three internal comparators and comparator A or comparator B or comparator P compare match functions, it consequently has three internal interrupts. 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

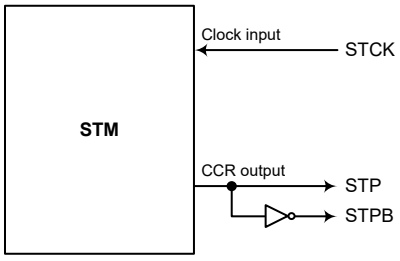
Each of the TMs, irrespective of what type, has one TM input pins, with the label xTCKn. The xTMn input pin, xTCKn, is essentially a clock source for the xTMn and is selected using the xTnCK2~xTnCK0 bits in the xTMnC0 register. This external TM input pin allows an external clock source to drive the internal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

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

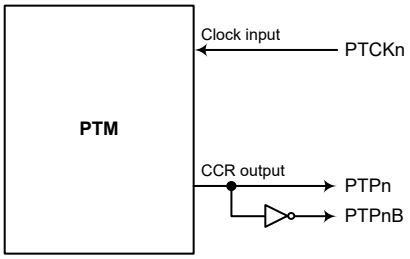
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using relevant pin-shared function selection register. The details of the pin-shared function selection are described in the pin-shared function section.

STM		PTMn		ATM	
Input	Output	Input	Output	Input	Output
STCK	STP, STPB	PTCK0, PTCK1	PTP0, PTP0B PTP1, PTP1B	ATCK	ATP/ATP_PWM1 ATPB/ATP_PWM2

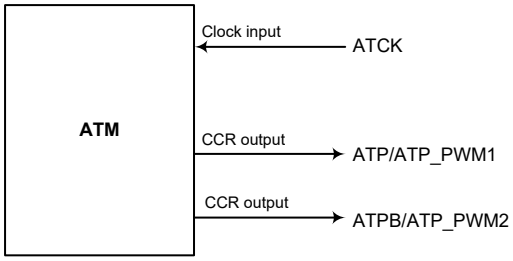
TM External Pins



STM Function Pin Block Diagram



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

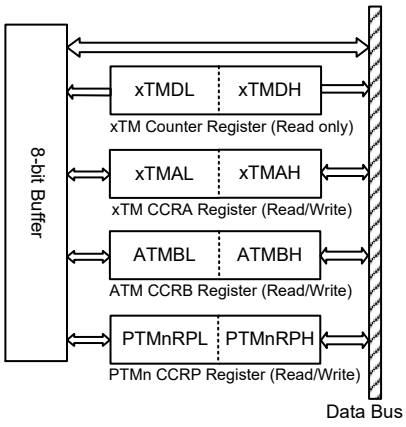


ATM Function Pin Block Diagram

Programming Considerations

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

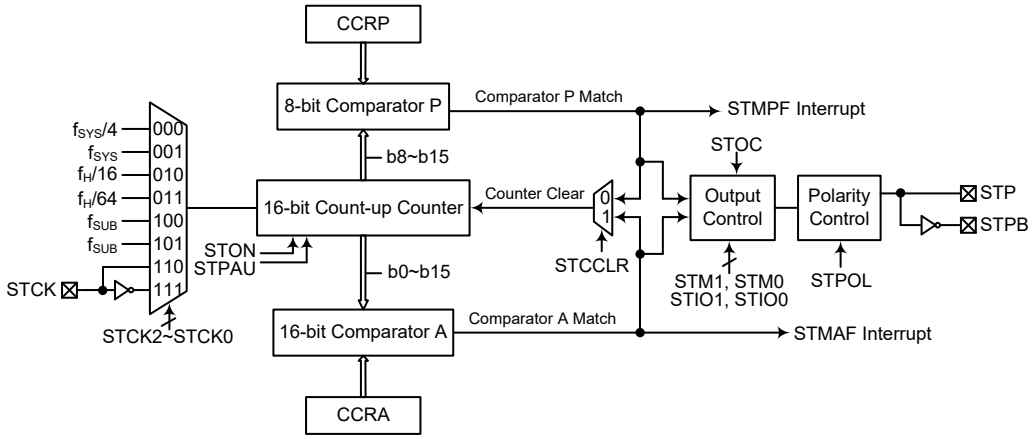


The following steps show the read and write procedures:

- Writing Data to CCRA or CCRB or CCRP
  - Step 1. Write data to Low Byte xTMnAL, ATMBL or PTMnRPL
    – Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte xTMnAH, ATMBH or PTMnRPH
    – Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA, CCRB or CCRP
  - Step 1. Read data from the High Byte xTMnDH, xTMnAH, ATMBH 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, ATMBL or PTMnRPL
    – This step reads data from the 8-bit buffer.

Standard Type TM – STM

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



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

16-bit Standard Type TM Block Diagram

Standard TM Operation

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

The only way of changing the value of the 16-bit counter using the application program, is to clear the counter by changing the STON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators.

When these conditions occur, a STM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including external input pins and can also control more than one 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 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
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 Type 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**: Select STM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: STCK rising edge clock
111: STCK falling edge clock
These three bits are used to select the clock source for the STM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the “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.

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

Bit 2~0 Unimplemented, read as "0"

• STMC1 Register

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

Bit 7~6 **STM1~STM0**: Select STM Operating Mode
00: Compare Match Output Mode
01: Undefined
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**: Select STM external pin function

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

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

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. These two bits have no effect if the STM is in the Timer/Counter Mode.

In the Compare Match Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a compare match occurs from the Comparator A. The TM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both 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 STP output pin when the STON bit changes from low to high.

Bit 2 **STPOL**: 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 or Single Pulse Output 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 set to zero. 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 four operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

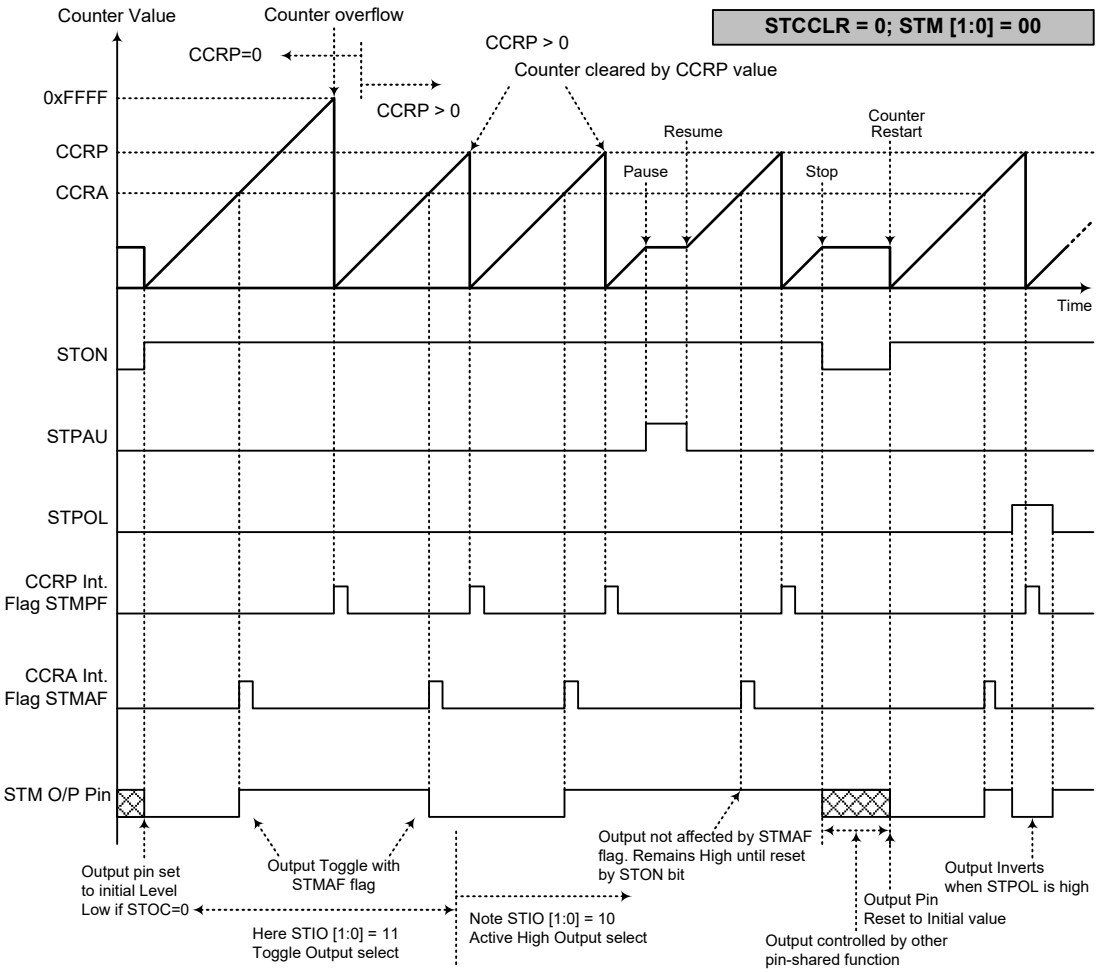
Compare Match Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register, should be set to 00. 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 zero.

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

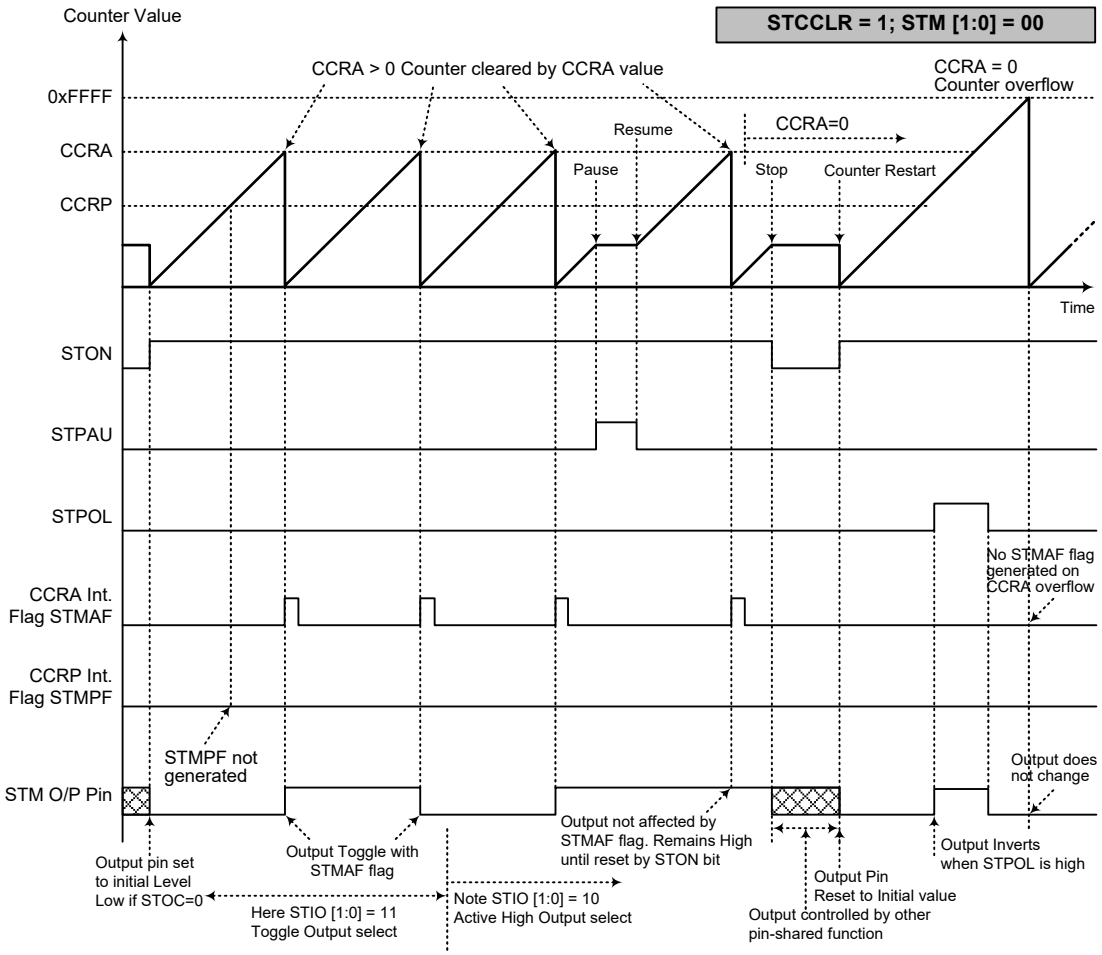


Compare Match Output Mode – STCCLR=0

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 its initial state by a STON bit rising edge



Compare Match Output Mode – STCCLR=1

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

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

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

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

Timer/Counter Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 11. 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. The PWM function within the STM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the STM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

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

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

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

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

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

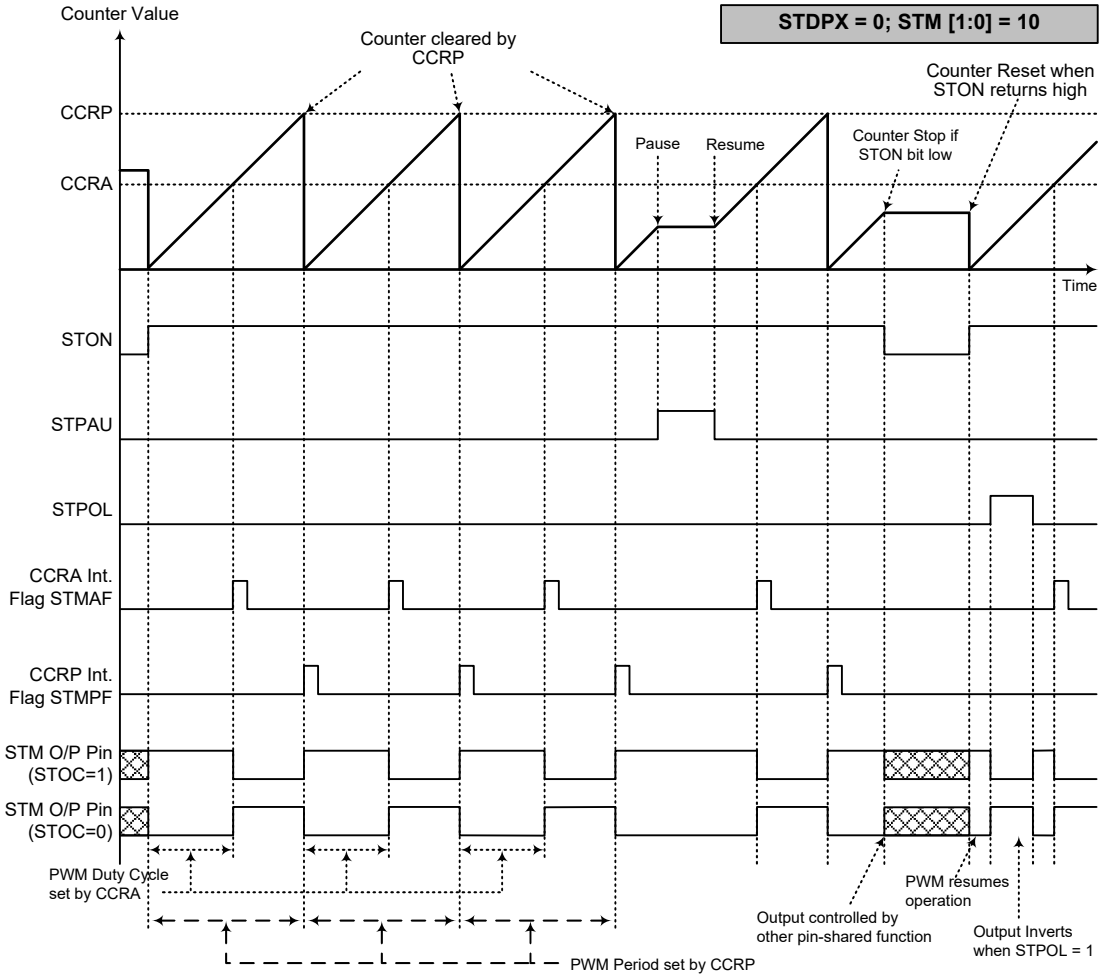
The STM PWM output frequency=(f_{SYS}/4)/(2×256)=f_{SYS}/2048=4kHz, duty=128/(2×256)=25%.

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.



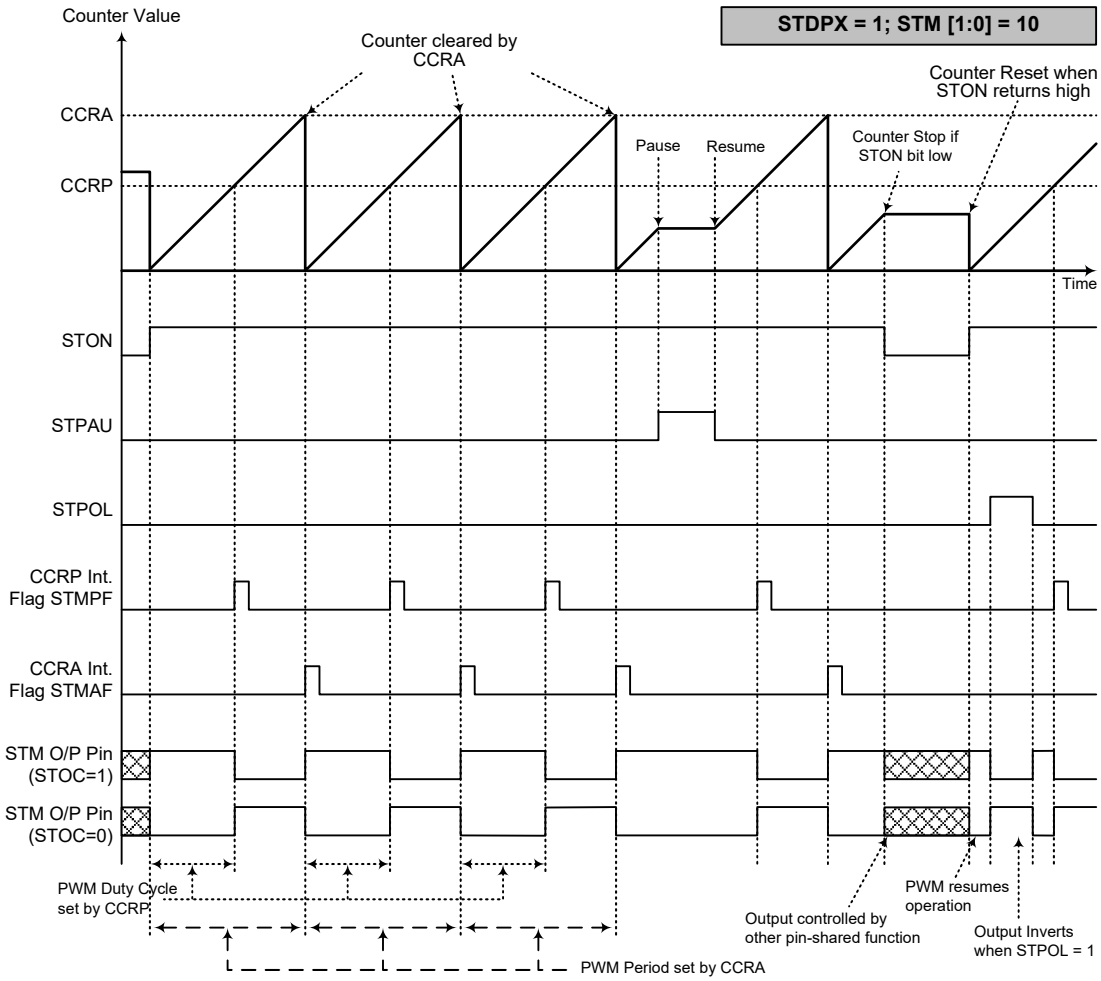
PWM Output Mode – STDPX=0

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

2. A counter clear sets 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 PWM Period

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

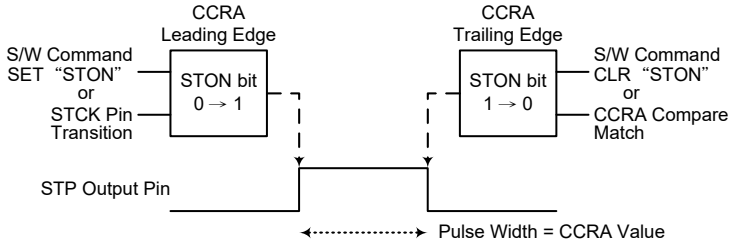
4. The STCCLR bit has no influence on PWM operation

Single Pulse Output Mode

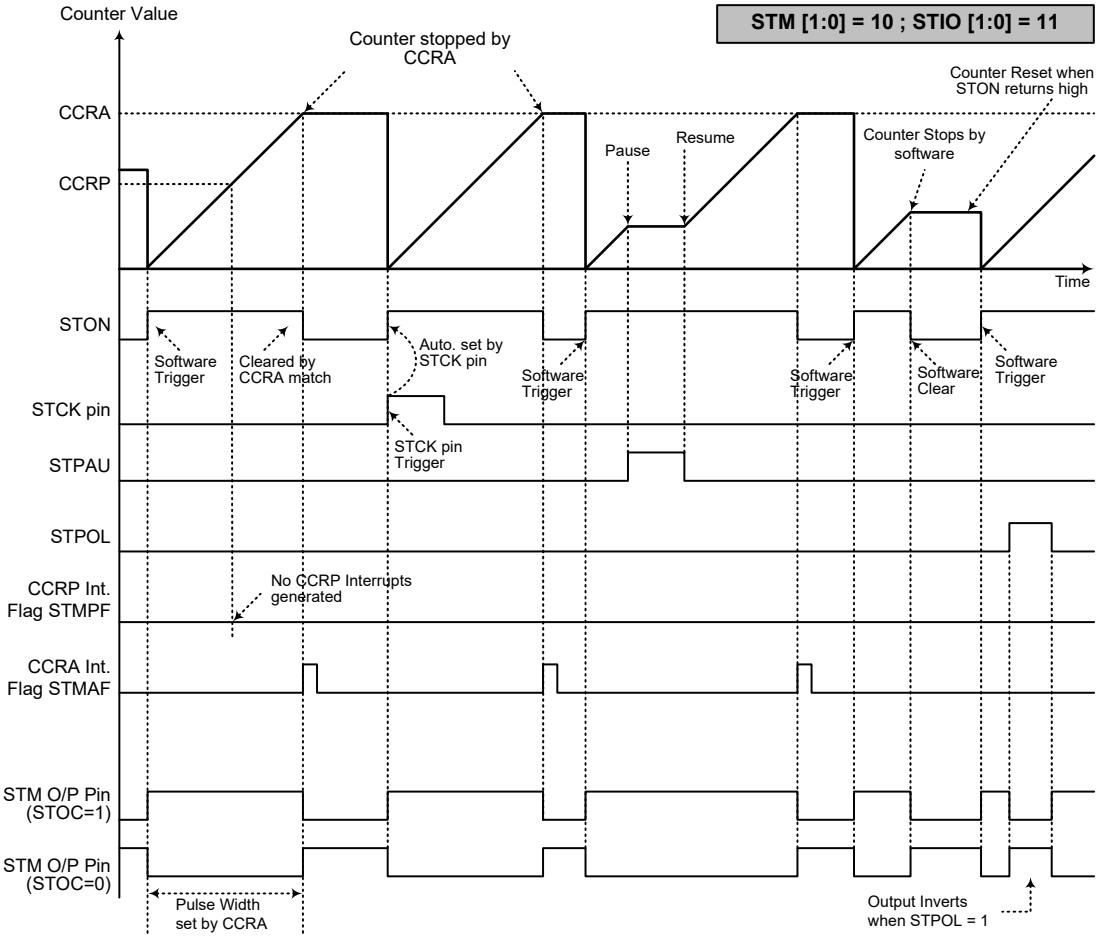
To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 and also the STIO1 and STIO0 bits should be set to 11. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

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 a 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

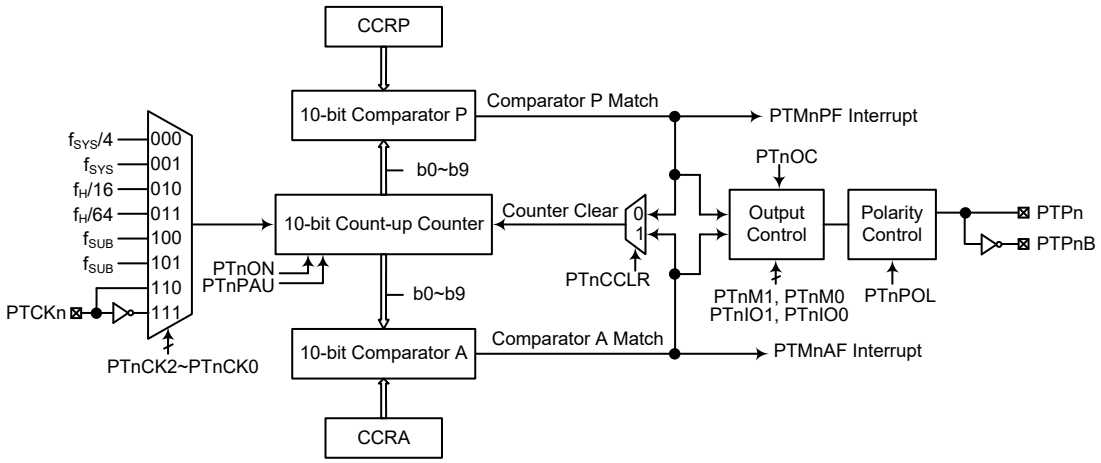
3. The pulse triggered by the STCK pin or by setting the STON bit high

4. A 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 can not be changed

Periodic Type TM – PTM

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



Note: The PTMn external pins are pin-shared with other functions, therefore before using the PTMn function, ensure that the pin-shared function registers have been set properly to enable the PTMn pin function. The PTCKn 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=0~1)

Periodic 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 an output pin. 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	D1	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

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

- **PTMnC0 Register**

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

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

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

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

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

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	D1	PTnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PTnM1~PTnM0**: Select PTMn Operating Mode
00: Compare Match Output Mode
01: Undefined
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**: Select PTMn external pin function

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

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

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. These two bits have no effect if the PTMn is in the Timer/Counter Mode.

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 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 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/Single Pulse Output Mode it determines if the PWM signal is active high or active low.

Bit 2 **PTnPOL**: PTMn PTPn Output polarity control
0: Non-invert
1: Invert
This bit controls the polarity of the PTPn output pin. When the bit is set high the PTMn output pin will be inverted and not inverted when the bit is zero. It has no effect if the PTMn is in the Timer/Counter Mode.

Bit 1 **D1**: Reserved bit, must be fixed at "0"

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 or Single Pulse Output 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 four operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode or Timer/Counter Mode. The operating mode is selected using the PTnM1 and PTnM0 bits in the PTMnC1 register.

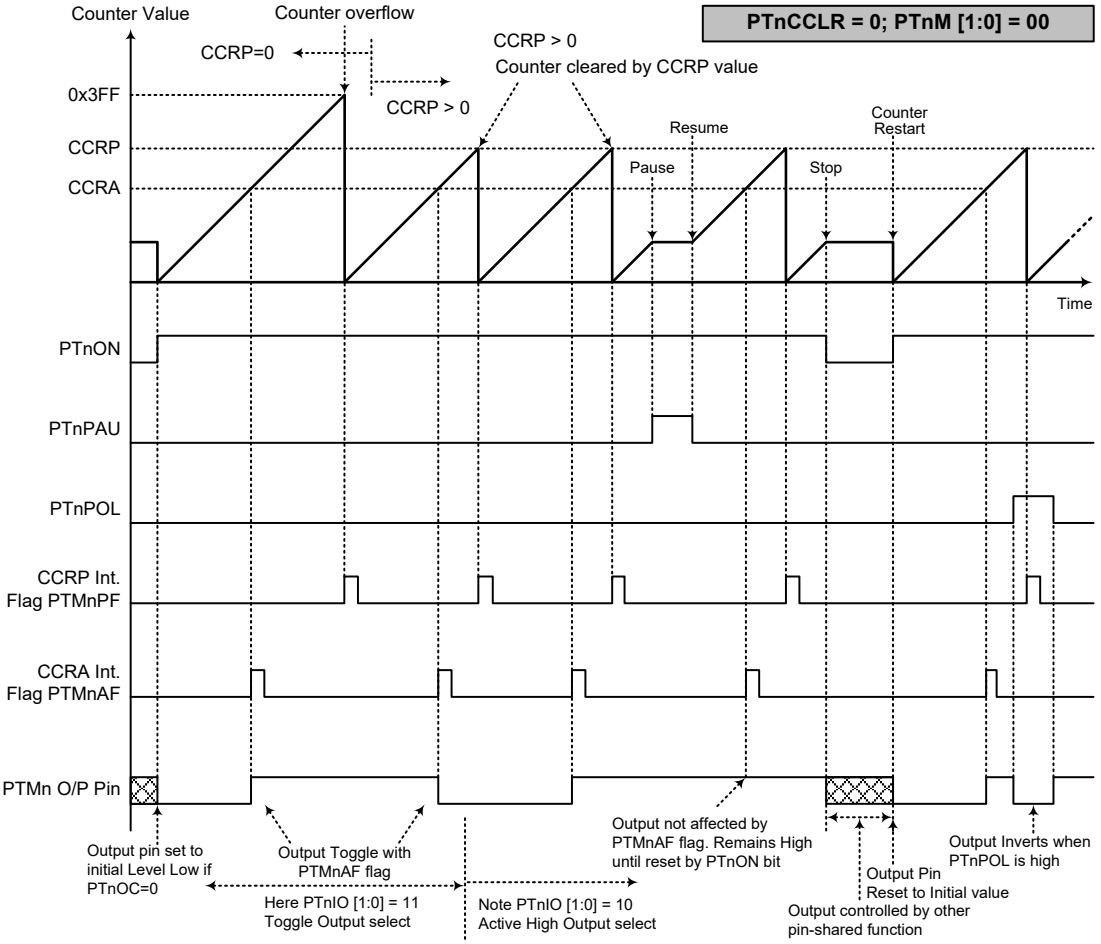
Compare Match Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register, should be set to 00. 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 set to "0".

If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the 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.

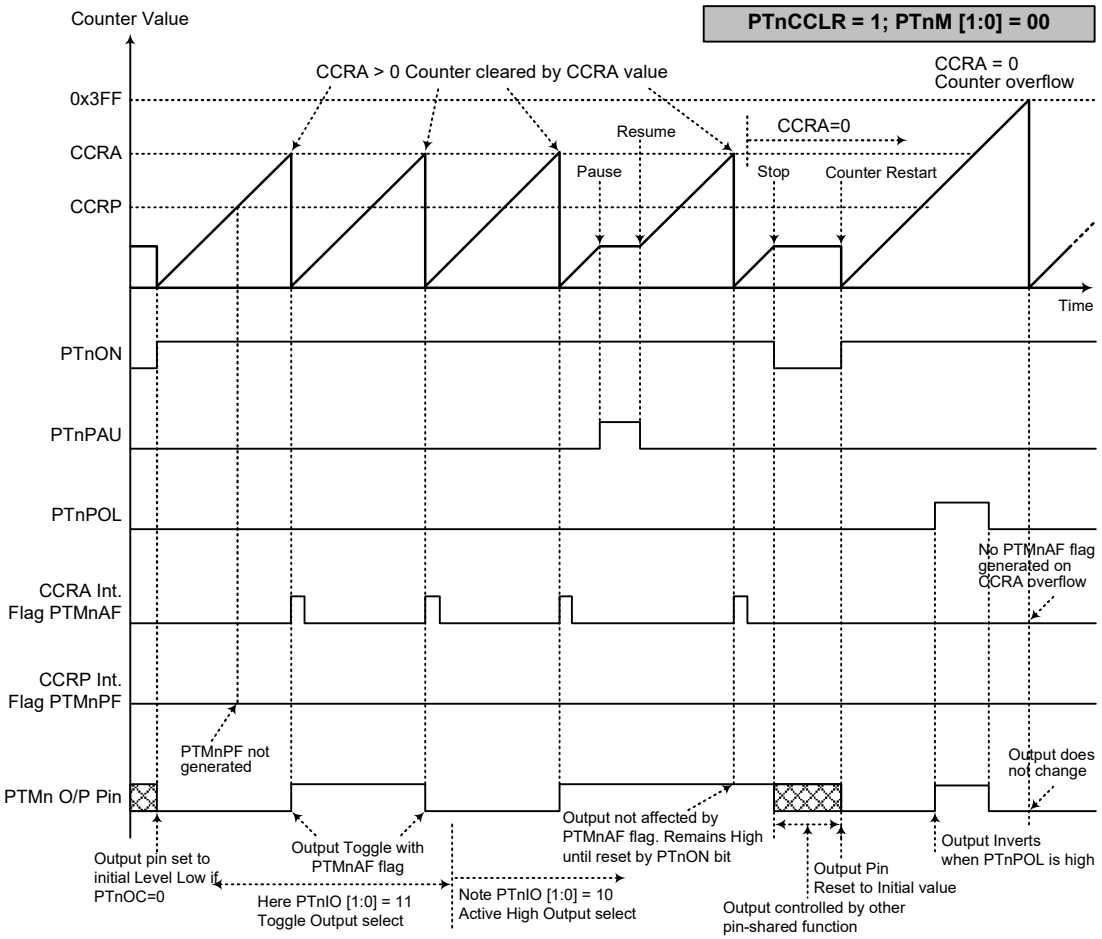


Compare Match Output Mode – PTnCCLR=0

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

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

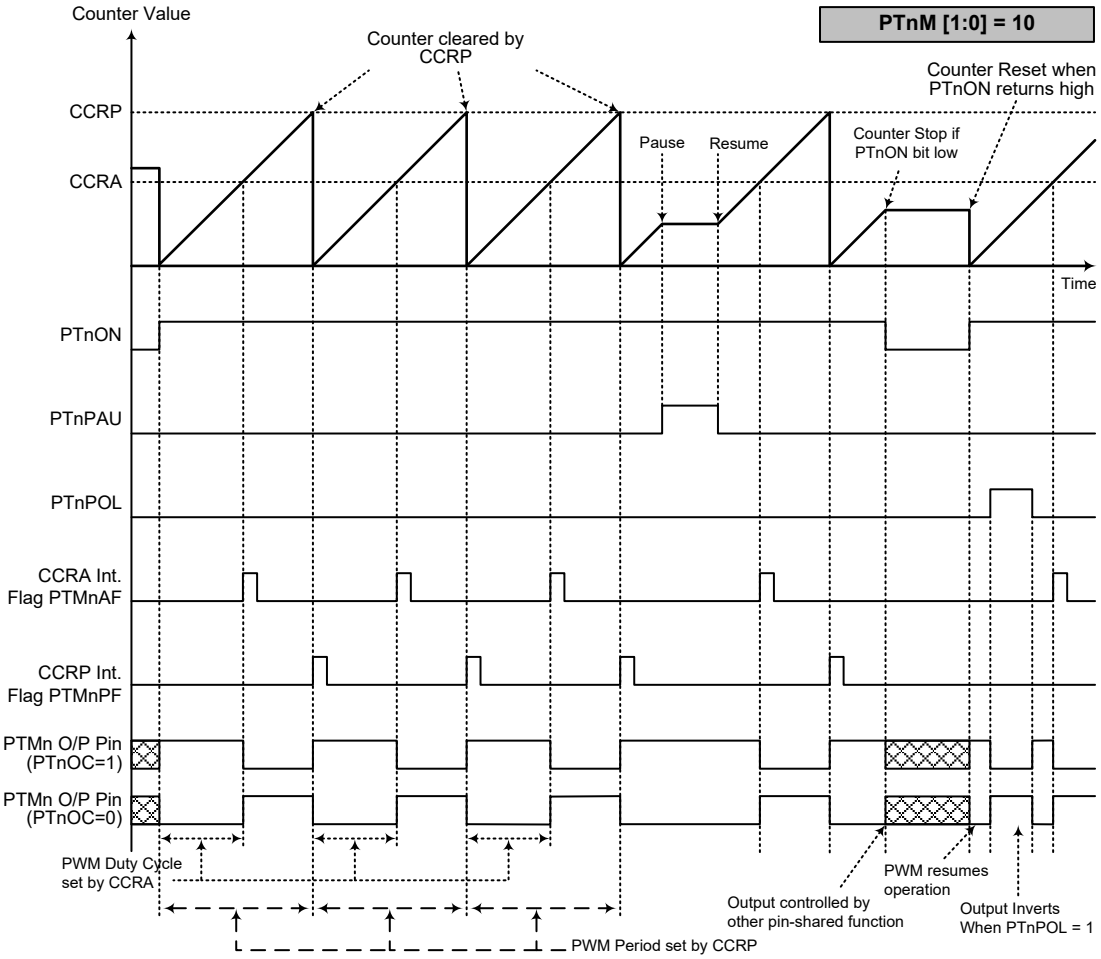
- **10-bit PTMn, PWM Output Mode, Edge-aligned Mode**

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

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

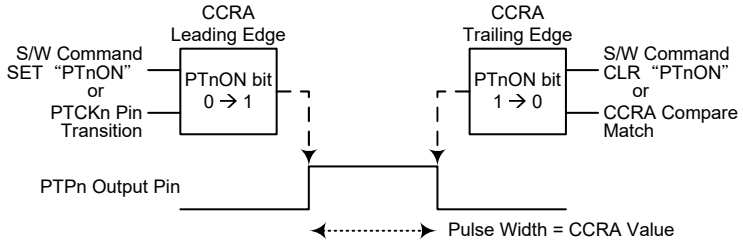
4. The PTnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

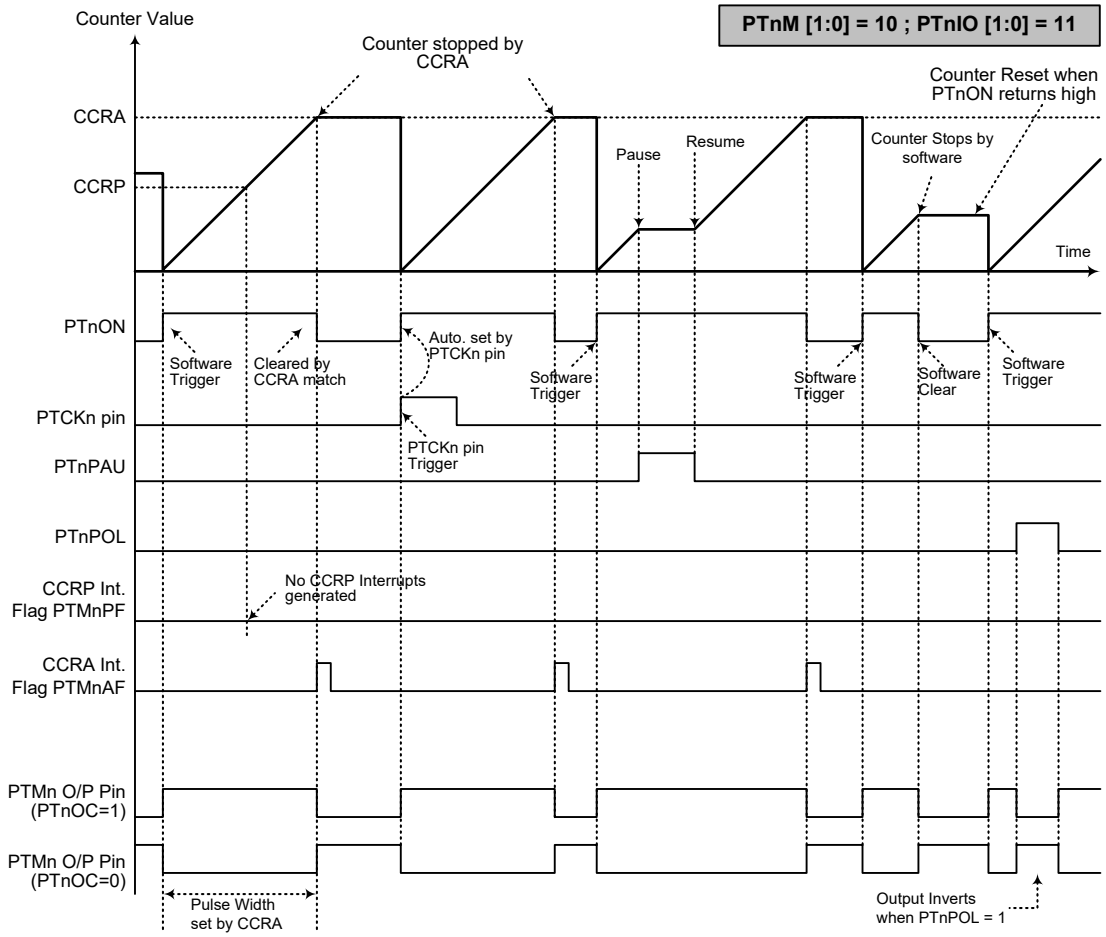
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 and also the PTnIO1 and PTnIO0 bits should be set to 11. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

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 bit is not used in this Mode.



Single Pulse Generation



Single Pulse Output Mode

Note: 1. Counter stopped by CCRA

2. CCRP is not used

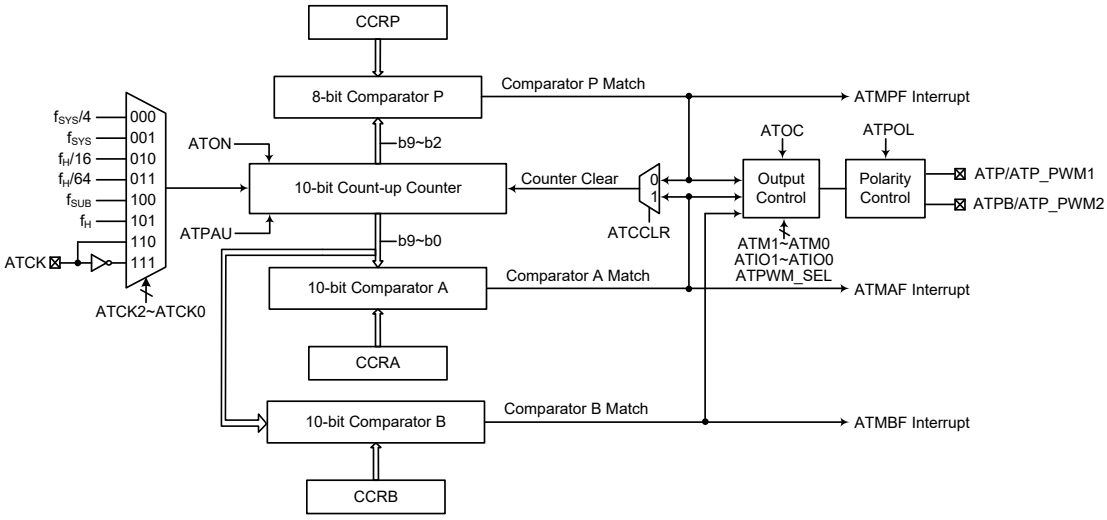
3. The pulse triggered by the PTCKn pin or by setting the PTnON bit high

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.

Audio Type TM – ATM

The Audio Type TM contains three operating modes which are Compare Match Output, Timer/ Event Counter and PWM Output modes. Here the PWM Output mode contains two sub-modes, namely Normal PWM Output mode and Audio PWM Output mode. The Audio type TM can also be controlled with an external input pin and can drive one or more external outputs. These output signals can be the same signal or the inverse signals. When the Audio PWM Output mode is enabled, the Audio PWM outputs will be generated on the ATP_PWM1 and ATP_PWM2 pins and can be used to drive speakers through an external PWM driver.



Note: 1. The ATM external pins are pin-shared with other functions, therefore before using the ATM function, ensure that the pin-shared function registers have been properly set to enable the ATM pin function. Additionally the ATCK pin, if used, must also be set as an input by setting the corresponding bit in the port control register.

2. Only in the Audio PWM Output mode, will the ATP_PWM1 and ATP_PWM2 pin functions be used. In other ATM modes including Compare match output mode and Normal PWM Output mode, the pins are used as ATP and ATPB functions. The ATPB is inverted signal of the ATP output.

10-bit Audio Type TM Block Diagram

Audio TM Operation

At its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are three internal comparators with the names, Comparator A, Comparator B and Comparator P. These comparators will compare the value in the counter with the CCRA, CCRB and CCRP registers. The CCRP comparator is 8-bit wide whose value is compared with the highest 8-bits in the counter while the CCRA and CCRB comparators are 10-bit wide and therefore 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 ATON 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 ATM interrupt signal will also usually be generated. The Audio 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 one or more output pins. All operating setup conditions are selected using relevant internal registers.

Audio Type TM Register Description

Overall operation of the Audio Type TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10-bit value, while two read/write register pairs exist to store the internal 10-bit CCRA and CCRB values. The ATMRP register is used to store the 8-bit 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
ATMC0	ATPAU	ATCK2	ATCK1	ATCK0	ATON	—	—	ATPWM_SEL
ATMC1	ATM1	ATM0	ATIO1	ATIO0	ATOC	ATPOL	ATDPX	ATCCLR
ATMDL	D7	D6	D5	D4	D3	D2	D1	D0
ATMDH	—	—	—	—	—	—	D9	D8
ATMAL	D7	D6	D5	D4	D3	D2	D1	D0
ATMAH	—	—	—	—	—	—	D9	D8
ATMBL	D7	D6	D5	D4	D3	D2	D1	D0
ATMBH	—	—	—	—	—	—	D9	D8
ATMRP	D7	D6	D5	D4	D3	D2	D1	D0

10-bit Audio Type TM Register List

• ATMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	ATPAU	ATCK2	ATCK1	ATCK0	ATON	—	—	ATPWM_SEL
R/W	R/W	R/W	R/W	R/W	R/W	—	—	R/W
POR	0	0	0	0	0	—	—	0

Bit 7 **ATPAU**: ATM 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 ATM 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 **ATCK2~ATCK0**: Select ATM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_H
110: ATCK rising edge clock
111: ATCK falling edge clock

These three bits are used to select the clock source for the ATM. 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 **ATON**: ATM Counter On/Off control
0: Off
1: On

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

Bit 2~1 Unimplemented, read as "0"

Bit 0 **ATPWM_SEL**: PWM Output Mode selection
0: Normal PWM Output Mode
1: Audio PWM Output Mode

This bit controls the PWM Output Mode selection. When the PWM output mode has been selected using the relevant bits in the ATMC1 register, then setting this bit high will switch the ATM operation from Normal PWM Output mode to the Audio PWM Output mode.

• ATMC1 Register

Bit	7	6	5	4	3	2	1	0
Name	ATM1	ATM0	ATIO1	ATIO0	ATOC	ATPOL	ATDPX	ATCCLR
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 **ATM1~ATM0**: Select ATM Operating Mode
00: Compare Match Output Mode
01: Undefined
10: PWM Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the ATM. To ensure reliable operation the ATM should be switched off before any changes are made to the ATM1 and ATM0 bits. In the Timer/Counter Mode, the ATM output pin state is undefined.

Bit 5~4 **ATIO1~ATIO0**: Select ATM external pin function

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

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

These two bits are used to determine how the ATM external pin changes state when a certain condition is reached. The function that these bits select depends upon the mode in which the ATM is running. These two bits have no effect if the ATM is in the Timer/Counter Mode.

In the Compare Match Output Mode, the ATIO1 and ATIO0 bits determine how the ATM output pin changes state when a compare match occurs from Comparator A. The ATM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the ATM output pin should be setup using the ATOC bit in the ATMC1 register. Note that the output level requested by the ATIO1 and ATIO0 bits must be different from the initial value setup using the ATOC bit otherwise no change will occur on the ATM output pin when a compare match occurs. After the ATM output pin changes state, it can be reset to its initial level by changing the level of the ATON bit from low to high.

In the PWM Output Mode, the ATIO1 and ATIO0 bits determine how the ATP or the ATP_PWM1 and ATP_PWM2 outputs change 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 ATIO1 and ATIO0 bits only after the ATM has been switched off. Unpredictable PWM outputs will occur if the ATIO1 and ATIO0 bits are changed when the ATM is running.

Bit 3 **ATOC**: ATM output pin 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 ATM output pins. Its operation depends upon whether the ATM is being used in the Compare Match Output Mode or in the PWM Output Mode. It has no effect if the ATM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the ATP output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signals of the ATP or the ATP_PWM1 and ATP_PWM2 are active high or active low.

Bit 2 **ATPOL**: ATM output polarity control
0: Non-invert
1: Invert

This bit controls the polarity of the ATM output. When the bit is set high the ATP or the ATP_PWM1 and ATP_PWM2 outputs will be inverted and not inverted when the bit is zero. It has no effect if the ATM is in the Timer/Counter Mode.

Bit 1 **ATDPX**: ATM PWM period/duty control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

This bit determines which register of the CCRA and CCRP is used to control either frequency or duty cycle of the PWM waveform in the Normal PWM Output Mode. However if the Audio PWM Output Mode is selected by setting the ATPWM_SEL bit high, the ATDPX bit will be cleared to zero by hardware which means that the period values of output waveforms on the ATP_PWM1 and ATP_PWM2 will be controlled by the CCRP while their duty cycle values are controlled by the CCRA and CCRB.

Bit 0 **ATCCLR**: ATM 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 there are two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the ATCCLR bit set high, the counter will be cleared when a compare match occurs from Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from 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 ATCCLR bit is not used in the PWM Output Mode.

- **ATMDL 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**: ATM Counter Low Byte Register bit 7 ~ bit 0
ATM 10-bit Counter bit 7 ~ bit 0

• **ATMDH 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**: ATM Counter High Byte Register bit 1 ~ bit 0
ATM 10-bit Counter bit 9 ~ bit 8

• **ATMAL 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**: ATM CCRA Low Byte Register bit 7 ~ bit 0
ATM 10-bit CCRA bit 7 ~ bit 0

• **ATMAH 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**: ATM CCRA High Byte Register bit 1 ~ bit 0
ATM 10-bit CCRA bit 9 ~ bit 8

• **ATMBL 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**: ATM CCRB Low Byte Register bit 7 ~ bit 0
ATM 10-bit CCRB bit 7 ~ bit 0

• **ATMBH 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**: ATM CCRB High Byte Register bit 1 ~ bit 0
ATM 10-bit CCRB bit 9 ~ bit 8

• **ATMRP 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**: ATM CCRP 8-bit register, compared with the ATM Counter bit 9~bit 2

Comparator P Match Period

0: 1024 ATM clocks

1~255: (1~255) × 4 ATM 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 ATCCLR bit is cleared to zero. As the CCRP bits are only compared with the highest eight counter bits, the compare values exist in 4 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

Audio Type TM Operation Modes

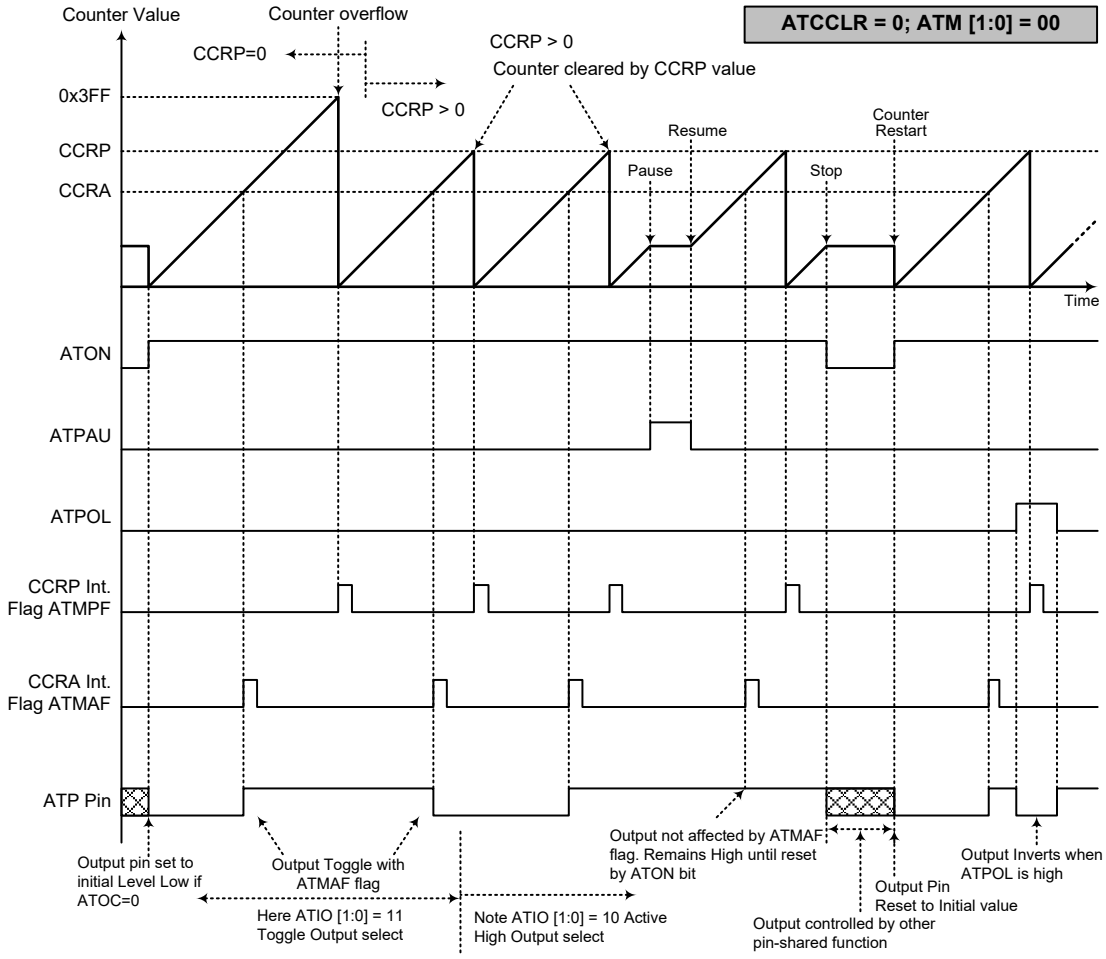
The Audio Type TM can operate in one of three operating modes, Compare Match Output Mode, Timer/Counter Mode and PWM Output Mode. The operating mode is selected using the ATM1 and ATM0 bits in the ATMC1 register.

Compare Match Output Mode

To select this mode, bits ATM1 and ATM0 in the ATMC1 register, should be set to 00. 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 ATCCLR 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 ATMAF and ATMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the ATCCLR bit in the ATMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the ATMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when ATCCLR is high no ATMPF 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 ATMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the ATM output pin will change state. The ATM output pin condition however only changes state when an ATMAF interrupt request flag is generated after a compare match occurs from Comparator A. The ATMPF interrupt request flag, generated from a compare match with Comparator P, will have no effect on the ATM output pin. The way in which the ATM output pin changes state are determined by the condition of the ATIO1 and ATIO0 bits in the ATMC1 register. The ATM output pin can be selected using the ATIO1 and ATIO0 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 ATM output pin, which is setup after the ATON bit changes from low to high, is setup using the ATOC bit. Note that if the ATIO1 and ATIO0 bits are zero then no pin change will take place.

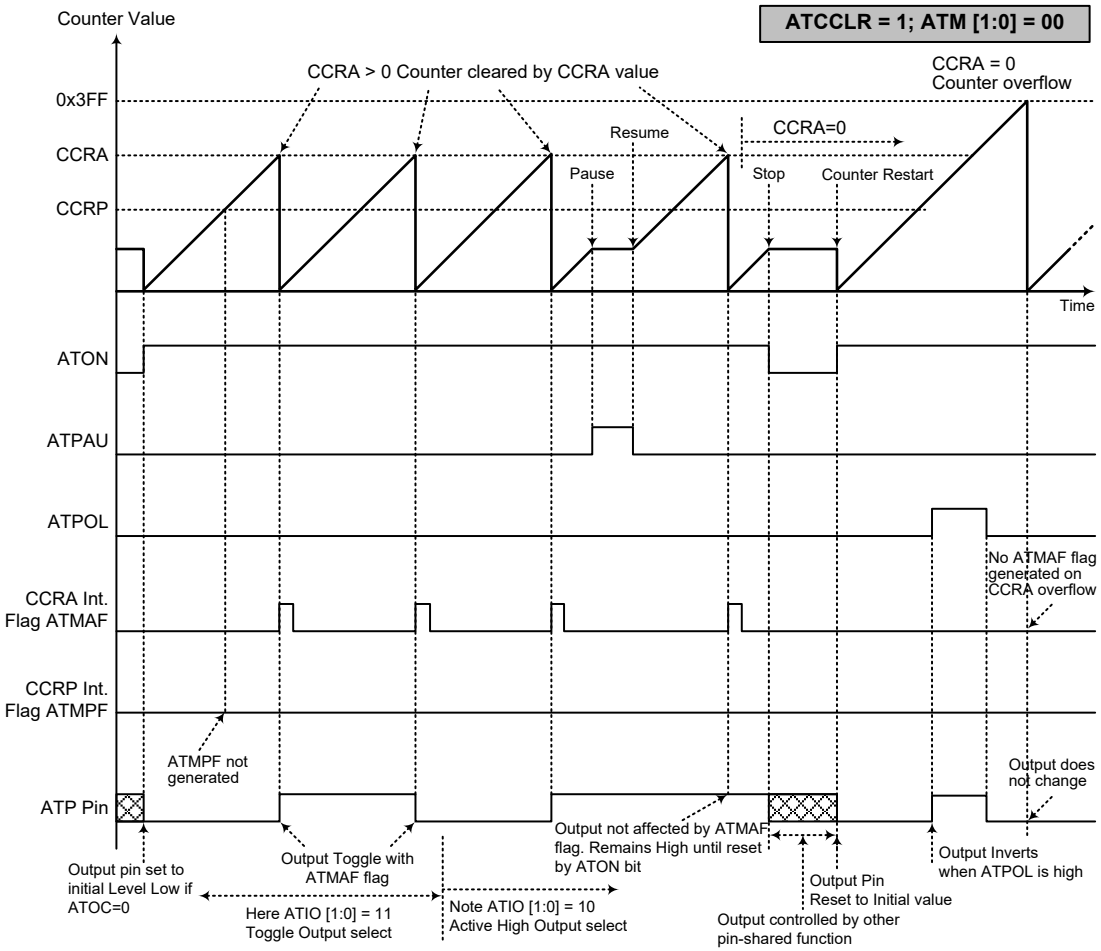


Compare Match Output Mode – ATCCLR=0

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

2. The ATM output pin is controlled only by the ATMAF flag

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



Compare Match Output Mode – ATCCLR=1

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

2. The ATM output pin is controlled only by the ATMAF flag
3. The output pin is reset to its initial state by a ATON bit rising edge
4. The ATMPF flag is not generated when ATCCLR=1

Timer/Counter Mode

To select this mode, bits ATM1 and ATM0 in the ATMC1 register should be set to 11. 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 ATM 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 ATM 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 ATM1 and ATM0 in the ATMC1 register should first be set to 10. The PWM function within the ATM is useful for applications which require functions such as a speaker driver, motor control, heating control, illumination control, etc. By providing a signal of fixed frequency but of varying duty cycle on the ATM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values. In the PWM Output Mode, 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 ATCCLR bit has no effect on the PWM operation.

For this device, the PWM output can operate in two modes which are Normal PWM Output Mode and Audio PWM Output Mode and are selected using the ATPWM_SEL bit in the ATMC0 register. The control bits and differences between the Normal PWM Output Mode and Audio PWM Output Mode are summarised in the accompanying table.

Item	Normal PWM Output Mode	Audio PWM Output Mode
ATPWM_SEL Bit	0	1
ATM CCRB Interrupt	×	√
Output Pins	ATP, ATPB	ATP_PWM1, ATP_PWM2
PWM Output Control	ATIO1~ATIO0 bits	
Output Active Level	ATOC bit	
Output Polarity Control	ATPOL bit	
Period/Duty Control	ATDPX bit	ATDPX bit is always 0. The Audio PWM period is controlled by CCRP.

Normal/Audio PWM Output Mode Comparison Table

Normal PWM Output Mode

In the Normal PWM Output Mode, 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 frequency or duty cycle is determined using the ATDPX bit in the ATMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The ATOC bit in the ATMC1 register is used to select the required polarity of the PWM waveform while the two ATIO1 and ATIO0 bits are used to enable the PWM output or to force the ATM output pin to a fixed high or low level. The ATPOL bit is used to reverse the polarity of the PWM output waveform.

• 10-bit ATM, Normal PWM Output Mode, Edge-aligned Mode, ATDPX=0

CCRP	1~255	0
Period	CCRP×4	1024
Duty	CCRA	

If f_{SYS}=8MHz, ATM clock source is $f_{SYS}/4$, CCRP=64 and CCRA=64,

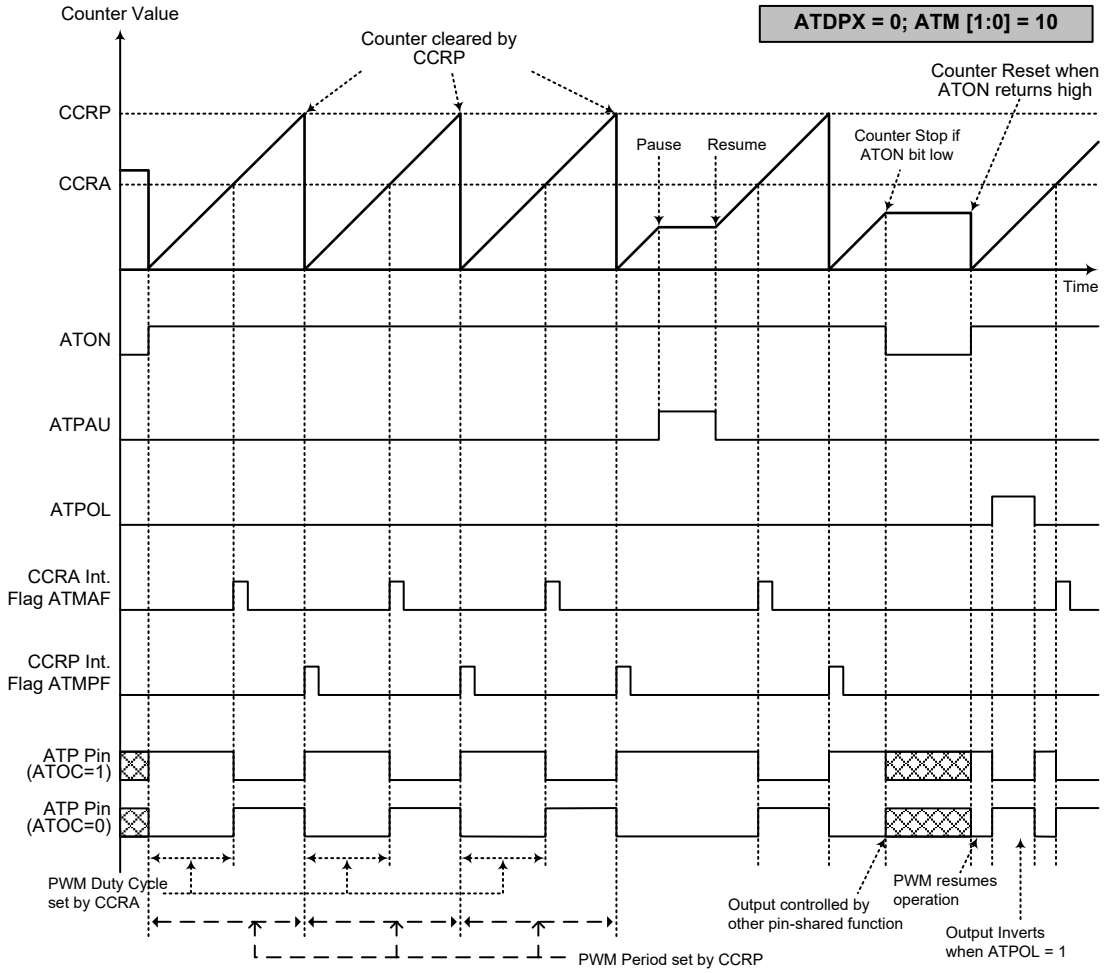
The ATM PWM output frequency=$(f_{SYS}/4)/(64\times4)=f_{SYS}/1024$=7.8125kHz, duty=64/(64×4)=25%,

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

• 10-bit ATM, Normal PWM Output Mode, Edge-aligned Mode, ATDPX=1

CCRP	1~255	0
Period	CCRA	
Duty	CCRP×4	1024

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



ATM Normal PWM Output Mode – ATDPX=0

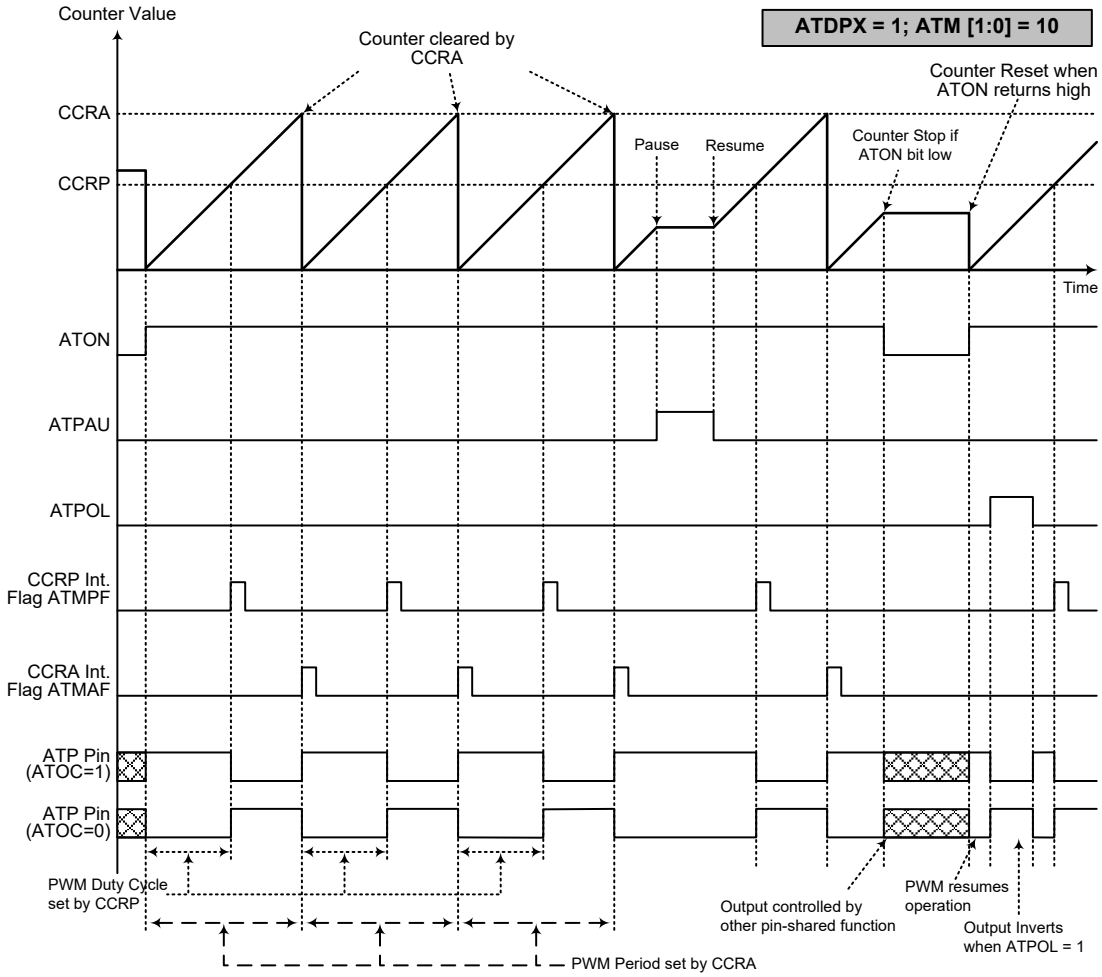
Note: 1. Here ATM[1:0]=10, ATPWM_SEL=0 and ATDPX=0

2. The counter is cleared by CCRP

3. A counter clear sets the PWM Period

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

5. The ATCCLR bit has no influence on PWM operation



ATM Normal PWM Output Mode – ATDPX=1

Note: 1. Here ATM[1:0]=10, ATPWM_SEL=0 and ATDPX=1

2. The counter is cleared by CCRA

3. A counter clear sets the PWM Period

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

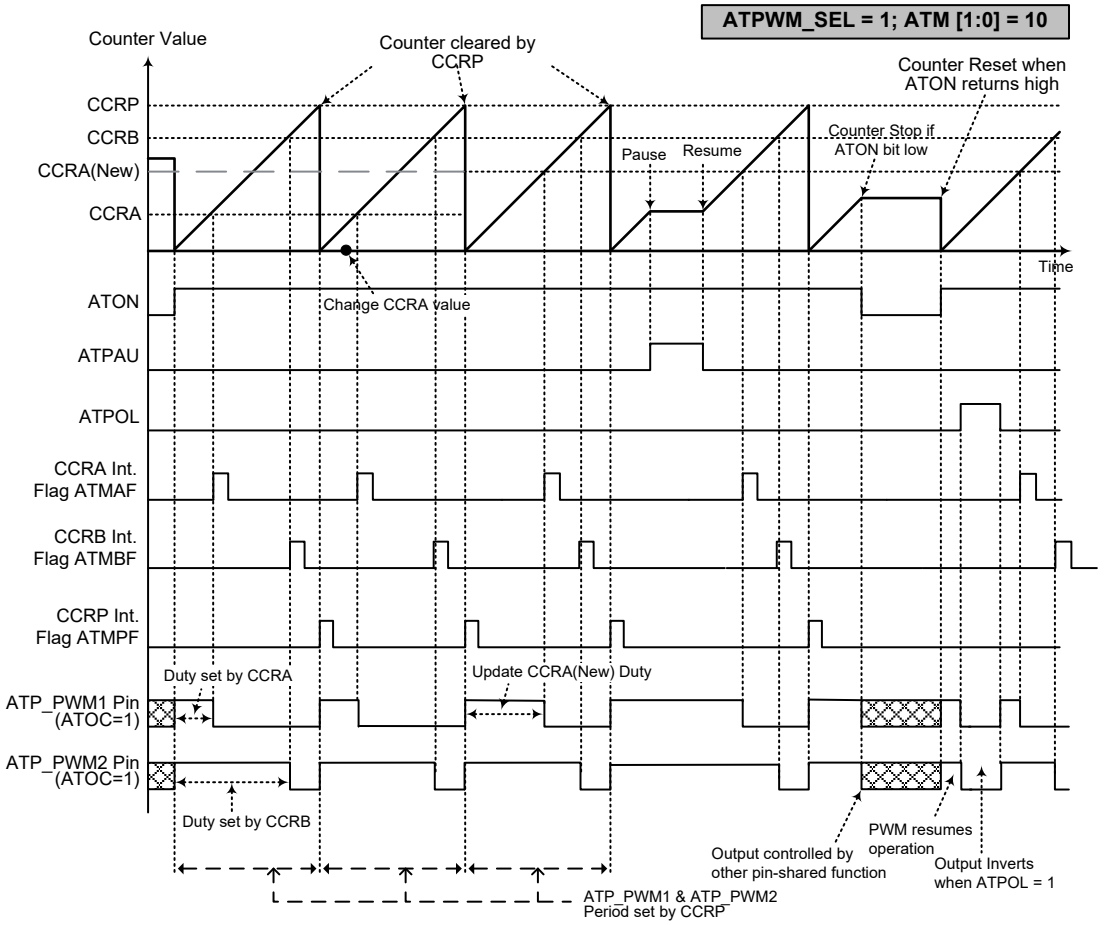
5. The ATCCLR bit has no influence on PWM operation

Audio PWM Output Mode

In the Audio PWM Output Mode, the CCRA, CCRB and CCRP registers are all used to generate the PWM waveforms on ATP_PWM1 and ATP_PWM2. The CCRP register is used to clear the internal counter and thus control the frequency of the two PWM outputs on the ATP_PWM1 and ATP_PWM2 pins, while the CCRA and CCRB registers are used to control the duty cycles of these two signals respectively.

An interrupt can be generated when a compare match occurs from either Comparator A, Comparator B or Comparator P. Except that the ATDPX bit is maintained at a zero value by hardware in the Audio PWM Output Mode, other control bits that control the ATP_PWM1 and ATP_PWM2 are the same as the Normal PWM Output mode.

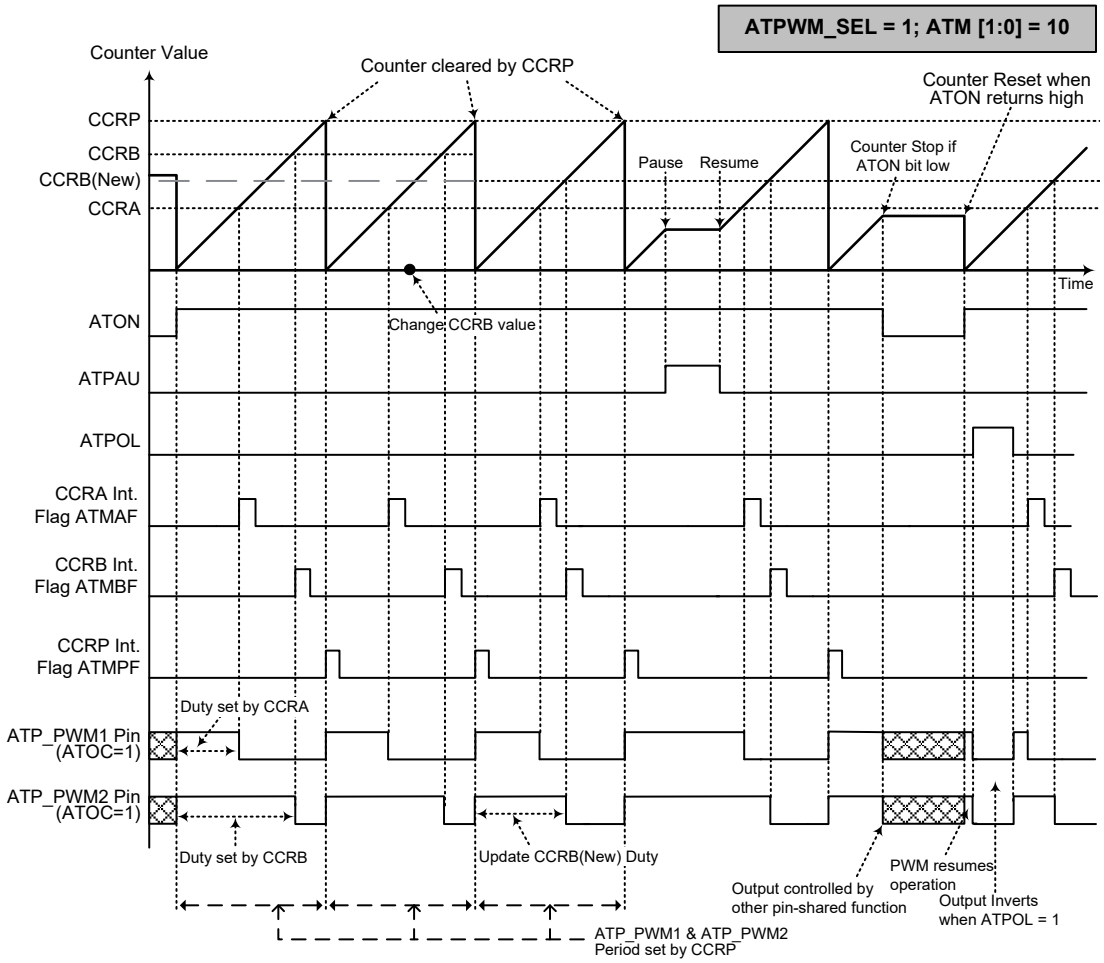
Note that as in the Audio PWM Output Mode, the CCRP, CCRA and CCRB all have a shadow function, when writting a new data into the CCRA, CCRB or CCRP registers, the CCRA, CCRB or CCRP will not be updated immediately by hardware until the current CCRP counter overflows and an ATMPF interrupt flag is set.



ATM Audio PWM Output Mode – Change CCRA Value

Note: 1. ATM[1:0]=10, ATPWM_SEL=1

2. The counter is cleared by CCRP
3. A counter clear sets the PWM Period
4. CCRA defines ATP_PWM1 duty while CCRB defines ATP_PWM2 duty
5. The internal PWM function continues running even when ATIO [1:0]=00 or 01
6. The ATCCLR bit has no influence on PWM operation, ATDPX bit is kept at 0 by hardware



ATM Audio PWM Output Mode – Change CCRB Value

Note: 1. ATM[1:0]=10, ATPWM_SEL=1

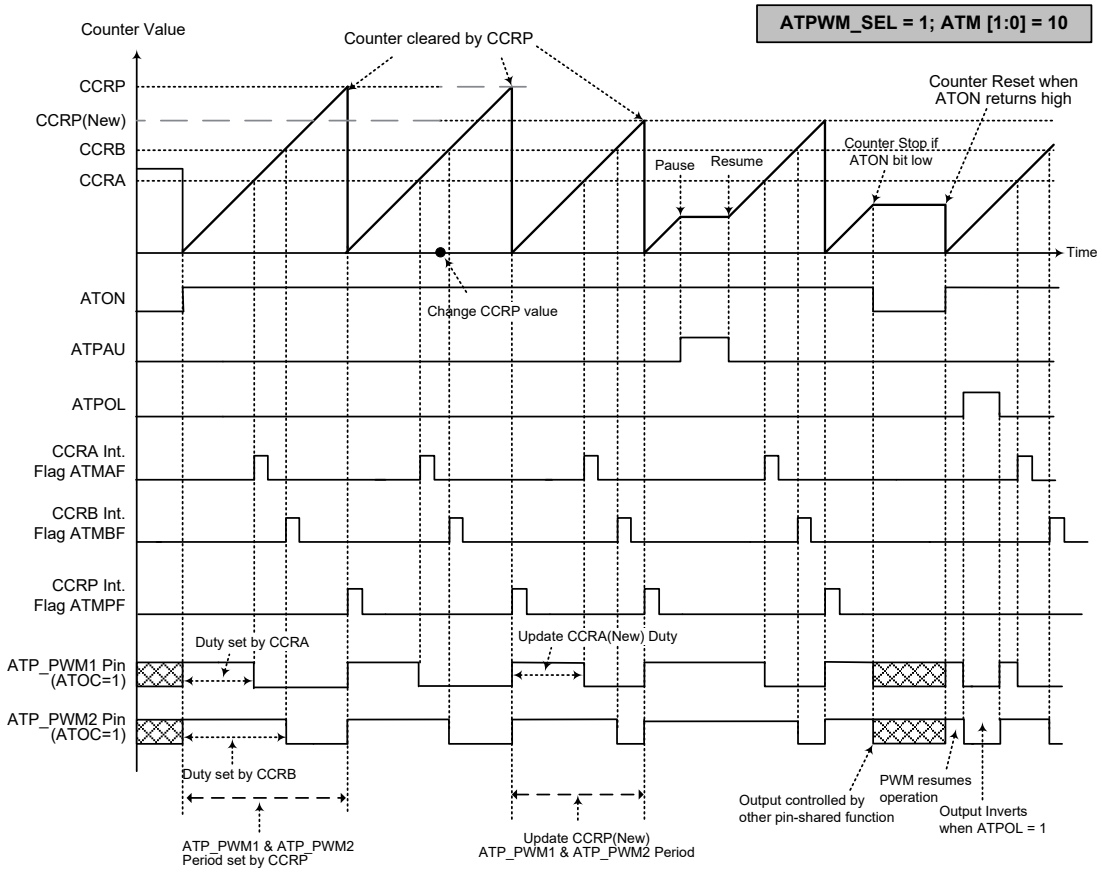
2. The counter is cleared by CCRP

3. A counter clear sets the PWM Period

4. CCRA defines ATP_PWM1 duty while CCRB defines ATP_PWM2 duty

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

6. The ATCCLR bit has no influence on PWM operation, ATDPX bit is kept at 0 by hardware



ATM Audio PWM Output Mode – Change CCRP Value

Note: 1. ATM[1:0]=10, ATPWM_SEL=1

2. The counter is cleared by CCRP

3. A counter clear sets the PWM Period

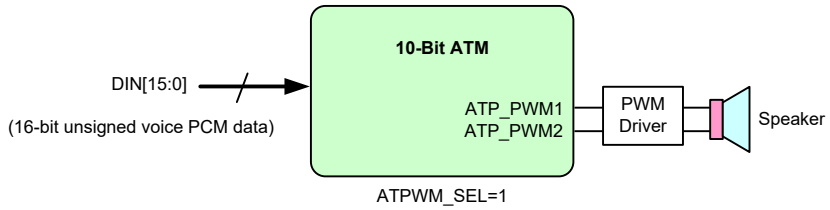
4. CCRA defines ATP_PWM1 duty while CCRB defines ATP_PWM2 duty

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

6. The ATCCLR bit has no influence on PWM operation, ATDPX bit is kept at 0 by hardware

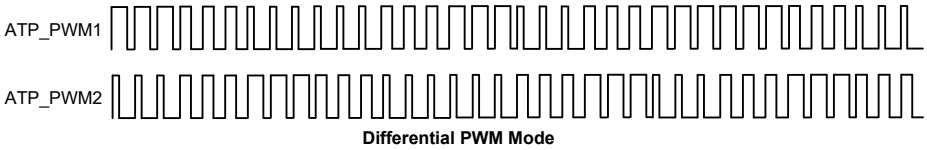
Audio PWM Output Usage Description

When operating in the Audio PWM Output Mode, the ATM is used to generate audio PWM waveforms according to decompressed PCM voice data that can then drive speakers through an external PWM driver. The following example introduces how to process 16-bit unsigned PCM voice data and write correct data into the CCRP, CCRA and CCRB registers to generate the corresponding audio PWM outputs.

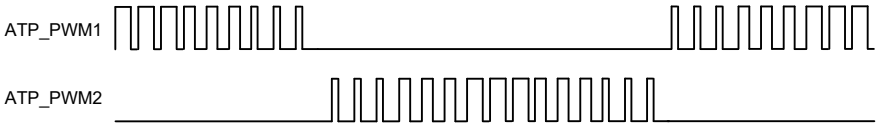


Audio PWM Output Application Schematic Diagram

The Audio PWM has two output modes, one is called the Differential Mode and the other is the Single Mode. Their output waveforms are illustrated in the following diagram.



Differential PWM Mode



Single PWM Mode

The following provides a description of how to use the ATM function in this device to generate the Differential and Single Audio PWM outputs. In this example, the PWM resolution is 8-bit and the system clock is derived from f_H which is 8MHz. To generated the following Audio PWM output signals, the CCRA, CCRB and CCRP register together with the relevant ATM control registers should be setup according to the steps below.

- Step 1 – Configure the ATMC0 Regsiter
  Select the ATM counter clock to be from either f_{SYS} or f_H by setting the ATCK2~ATCK0 bits in the ATMC0 register to "001" or "101". Set the ATPWM_SEL bit in the ATMC0 register to "1" to select the Audio PWM Output Mode. The ATMC0 register value will now be "00010--1b".
- Step 2 – Configure the ATMC1 Register
  Set the ATM1~ATM0 bits in the ATMC1 register to "10" and ATIO1~ATIO0 bits to "10" to ensure that the ATM is used with the PWM output. Then set the ATOC bit to "1" and ATPOL bit to "0" to select the PWM ouput to be active high and non-inverted. The ATMC1 register value will now be "10101000b".
- Step 3 – Configure the ATMRP Register
  Set the ATMRP register to "01000000b" to select the Comparator P match period as 256 ATM clocks. Therefore, the Audio PWM frequency=(f_{SYS})/256=31.25kHz. Of course, if the ATCK2~ATCK0 bits value is "000", the corresponding Audio PWM frequency is (f_{SYS}/4)/256=7.8125kHz.
- Step 4 – Pin-shared function selection
  Correctly set the PAS13~PAS10 bits in the PAS1 register or the PDS13~PDS10 bits in the PDS1 register to select the pin as corresponding ATP_PWM1 or ATP_PWM2 function.

- Step 5 – Enable the ATM function
  After setting up the other ATM related functions, such as interrupts, etc., then set the ATON bit in the ATMC0 register to "1" to enable the ATM counter to run.
- Step 6 – Update CCRA and CCRB Values
  According to the input data, the CCRA and CCRB values should be updated. Refer to the following section for the calculation of the CCRA and CCRB values.

As there are two types of Audio PWM output waveforms, Differential mode and Single mode, the calculation methods are different for these two types.

Example 1 – Audio PWM Output Differential Mode, DIN[15:0]=4000H

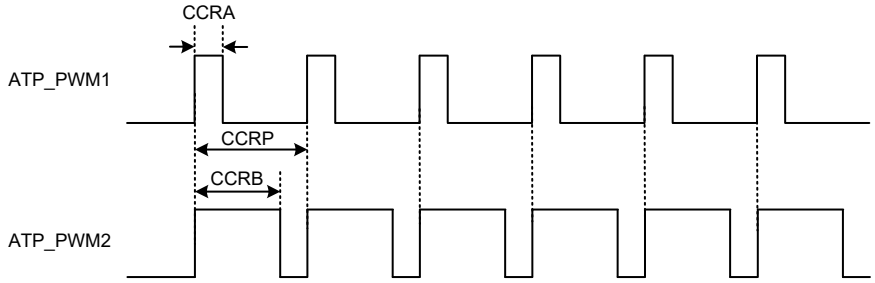
For example, DIN[15:0]=4000H (unsigned voice data), to generae the differential PWM outputs on ATP_PWM1 and ATP_PWM2 as shown in the timing diagram below,

DIN[15:0]=4000H, DINB[15:0]=~DIN[15:0]=BFFFH

The higher 8-bit of DIN: DIN[15:8]=40H=64, DINB[15:8]=BFH=191

For ATP_PWM1: 8-bit DIN[15:8]=40H=64=CCRA, Duty=64/256≈25%

For ATP_PWM2: 8-bit DINB[15:8]=BFH=191=CCRB, Duty=191/256≈75%



Example 2 – Audio PWM Output Differential Mode, DIN[15:0]=C000H

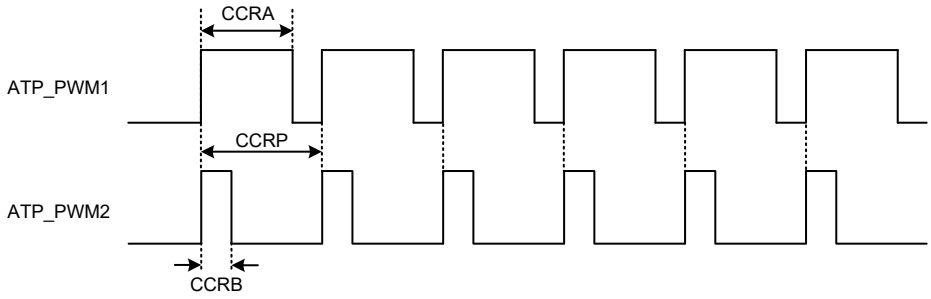
For example, DIN[15:0]=C000H (unsigned voice data), to generae the differential PWM outputs on ATP_PWM1 and ATP_PWM2 as shown in the timing diagram below,

DIN[15:0]=C000H, DINB[15:0]=~DIN[15:0]=3FFFH

The higher 8-bit of DIN: DIN[15:8]=C0H=192, DINB[15:8]=3FH=63

For ATP_PWM1: 8-bit DIN[15:8]=C0H=192=CCRA, Duty=192/256≈75%

For ATP_PWM2: 8-bit DINB[15:8]=3FH=63=CCRB, Duty=63/256≈25%



From the Example 1~Example 2 above, the calculation method for the CCRA/CCRB values converted from the DIN[15:0] unsigned voice data for Audio PWM Differential mode can be obtained.

CCRA=DIN[15:8]; CCRB=DINB[15:8]=~DIN[15:8]

The ATP_PWM1 and ATP_PWM2 output signal duty and the corresponding CCRA and CCRB values in the PWM output differential mode are shown in the following table.

DIN[15:0] (Unsigned Voice Data)	ATP_PWM1 Output	ATP_PWM2 Output	CCRA/CCRB Settings
(00H, 00H)	Pulse Duty≈0%	Pulse Duty≈100%	CCRA=DIN[15:8]=00H CCRB=~DIN[15:8]=FFH
(20H, 00H)	Pulse Duty≈12.5%	Pulse Duty≈87.5%	CCRA=DIN[15:8]=20H CCRB=~DIN[15:8]=DFH
(40H, 00H)	Pulse Duty≈25%	Pulse Duty≈75%	CCRA=DIN[15:8]=40H CCRB=~DIN[15:8]=BFH
(60H, 00H)	Pulse Duty≈37.5%	Pulse Duty≈62.5%	CCRA=DIN[15:8]=60H CCRB=~DIN[15:8]=9FH
(80H, 00H)	Pulse Duty≈50%	Pulse Duty≈50%	CCRA=DIN[15:8]=80H CCRB=~DIN[15:8]=7FH
(A0H, 00H)	Pulse Duty≈62.5%	Pulse Duty≈37.5%	CCRA=DIN[15:8]=A0H CCRB=~DIN[15:8]=5FH
(C0H, 00H)	Pulse Duty≈75%	Pulse Duty≈25%	CCRA=DIN[15:8]=C0H CCRB=~DIN[15:8]=3FH
(E0H, 00H)	Pulse Duty≈87.5%	Pulse Duty≈12.5%	CCRA=DIN[15:8]=E0H CCRB=~DIN[15:8]=1FH
(FFH, FFH)	Pulse Duty≈100%	Pulse Duty≈0%	CCRA=DIN[15:8]=FFH CCRB=~DIN[15:8]=00H

CCRA/CCRB Values – Audio PWM Output Differential Mode

Example 3 – Audio PWM Output Single Mode, DIN[15:0]=4000H

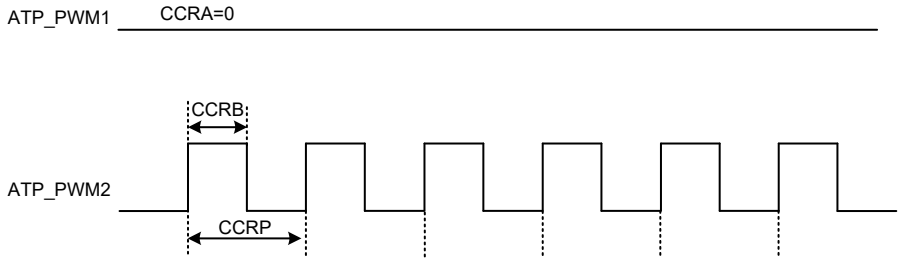
For example, DIN[15:0]=4000H (unsigned voice data), to generae the single PWM output on ATP_PWM1 or ATP_PWM2 as shown in the timing diagram below,

DIN[15:0]=4000H, DINB[15:0]=~DIN[15:0]=BFFFH

The Bit14~Bit 7 of DINB: DINB[14:7]=7FH=127

For ATP_PWM1: As DIN[15]=0, CCRA=0, Duty=0/256≈0%

For ATP_PWM2: DINB[14:7]=7FH=127=CCRB, Duty=127/256≈50%



Example 4- Audio PWM Output Single Mode, DIN[15:0]=C000H

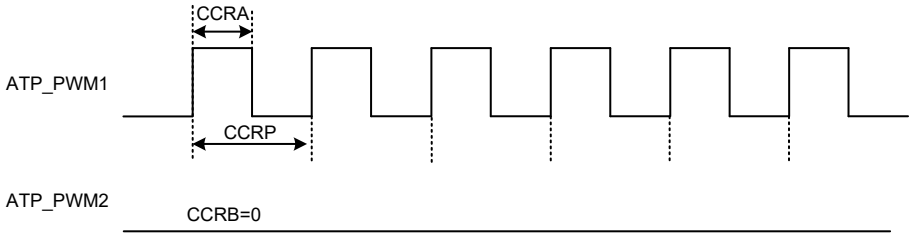
For example, DIN[15:0]=C000H (unsigned voice data), to generae the single PWM output on ATP_PWM1 or ATP_PWM2 as shown in the timing diagram below,

DIN[15:0]=C000H, DINB[15:0]=~DIN[15:0]=3FFFH

The Bit14~Bit7 of DIN: DIN[14:7]=80H=128

For the ATP_PWM1: DIN[14:7]=80H=128=CCRA, Duty=128/256≈50%

For the ATP_PWM2: As DIN[15]=1, CCRB=0, Duty=0/256≈0%



From the examples of Example 3~ 4 above, the calculation method for the CCRA/CCRB values converted from the DIN[15:0] unsigned voice data for the Audio PWM single mode can be obtained.

If DIN[15]=0, CCRA=0 (ATP_PWM1 outputs 0), CCRB=~DIN[14:7]

If DIN[15]=1, CCRA=DIN[14:7], CCRB=0 (ATP_PWM2 outputs 0)

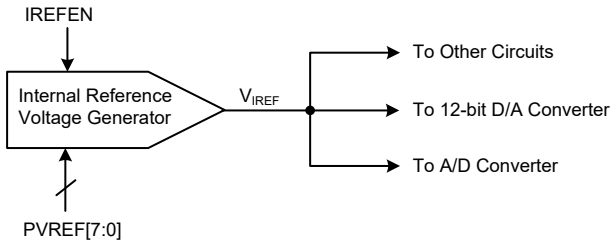
The ATP_PWM1 and ATP_PWM2 output signal duty and the corresponding CCRA and CCRB values in the PWM output single mode are shown in the following table.

DIN[15:0] (Unsigned Voice Data)	ATP_PWM1 Output	ATP_PWM2 Output	CCRA/CCRB Settings
(00H, 00H)	0	Pulse Duty≈100%	CCRA=00H CCRB=~DIN[14:7]=FFH
(20H, 00H)	0	Pulse Duty≈75%	CCRA=00H CCRB=~DIN[14:7]=BFH
(40H, 00H)	0	Pulse Duty≈50%	CCRA=00H CCRB=~DIN[14:7]=7FH
(60H, 00H)	0	Pulse Duty≈25%	CCRA=00H CCRB=~DIN[14:7]=3FH
(80H, 00H)	0	0	CCRA=DIN[14:7]=00H CCRB=00H
(A0H, 00H)	Pulse Duty≈25%	0	CCRA=DIN[14:7]=40H CCRB=00H
(C0H, 00H)	Pulse Duty≈50%	0	CCRA=DIN[14:7]=80H CCRB=00H
(E0H, 00H)	Pulse Duty≈75%	0	CCRA=DIN[14:7]=C0H CCRB=00H
(FFH, FFH)	Pulse Duty≈100%	0	CCRA=DIN[14:7]=FFH CCRB=00H

CCRA/CCRB Values- Audio PWM Output Single Mode

Internal Reference Voltage Generator

The device includes an internal reference voltage generator to provide an accurate reference voltage V_{IREF}. This reference voltage can be used as the internal 12-bit D/A Converter reference voltage or can be output through the DACVREF pin by properly settings to use for other circuits. Refer to the Internal Reference Voltage Characteristics section for more information.



Internal Reference Voltage Register Description

The internal reference voltage is controlled by two registers. The IREFC register is used for the enable/disable control while the PVREF register is used for fine tuning the internal reference voltage value.

Register Name	Bit							
	7	6	5	4	3	2	1	0
IREFC	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0
PVREF	D7	D6	D5	D4	D3	D2	D1	D0

Internal Reference Voltage Register List

• IREFC Register

Bit	7	6	5	4	3	2	1	0
Name	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0
R/W	R	R	—	R/W	R/W	—	R/W	R/W
POR	0	0	—	0	0	—	0	0

Bit 7 **OP2DO**: OPA2 digital output; positive logic
Refer to the Operational Amplifier section.

Bit 6 **OP1DO**: OPA1 digital output; positive logic
Refer to the Operational Amplifier section.

Bit 5 Unimplemented, read as "0"

Bit 4 **IREFEN**: Internal Reference Voltage Generator control
0: Disable
1: Enable
This bit controls the internal reference voltage generator on/off. When this bit is set high, the internal reference voltage V_{IREF} can be generated and be used as the D/A converter reference voltage. If the internal referenc voltage is not used by other circuits, it is recommended that the IREFEN bit is cleared to zero to reduce power consumption.

Bit 3 **BATDEN**: Battery Voltage Detection circuit control
0: Disable
1: Enable
This bit controls the battery voltage detection circuit on or off. When this bit is set high, the battery detection circuit is turned on and a $1/4V_{DD}$ voltage will be generated and then can be connected to the A/D converter input for being detected.

Bit 2 Unimplemented, read as "0"

Bit 1~0 **DACVRS1~DACVRS0**: 12-bit D/A Converter reference voltage $V_{DACVREF}$ selection
Refer to the D/A Converter Registers section.

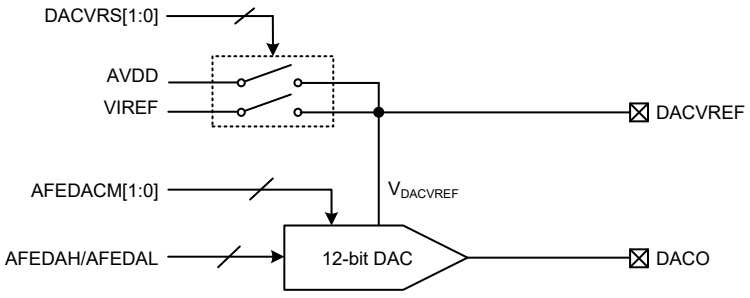
• PVREF Register

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**: Internal Reference Voltage Generator fine tune control
Setting the register can fine tune the internal reference voltage with a range of -60mV~+60mV (base on PVREF=80 (Hex)). When the PVREF register value is increased by one, the internal reference voltage will decrease around 500μV, vice verse.

Digital to Analog Converter – DAC

This device includes a 12-bit D/A Converter which can provide a certain voltage from 0 to $0.5\times V_{DACVREF}$ to the positive input of the internal operational amplifiers. The D/A converter output voltage can also be connected to the A/D converter input for measurement.



D/A Converter Structure

D/A Converter Register Description

The D/A converter overall function is controlled by four registers. The DACVRS1~DACVRS0 bits in the IREFC register are used to select the D/A converter reference voltage. The AFEDAC register is used for D/A converter function enable/disable control. A 12-bit value of the register pair, AFEDAH and AFEDAL, is used for the DAC output control.

Register Name	Bit							
	7	6	5	4	3	2	1	0
IREFC	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0
AFEDAC	—	—	—	—	—	—	AFEDACM1	AFEDACM0
AFEDAL	D3	D2	D1	D0	—	—	—	—
AFEDAH	D11	D10	D9	D8	D7	D6	D5	D4

D/A Converter Register List

• IREFC Register

Bit	7	6	5	4	3	2	1	0
Name	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0
R/W	R	R	—	R/W	R/W	—	R/W	R/W
POR	0	0	—	0	0	—	0	0

Bit 7 **OP2DO**: OPA2 digital output; positive logic
Refer to the Operational Amplifier section.

Bit 6 **OP1DO**: OPA1 digital output; positive logic
Refer to the Operational Amplifier section.

Bit 5 Unimplemented, read as "0"

Bit 4 **IREFEN**: Internal Reference Voltage Generator control
Refer to the Internal Reference Voltage Generator section.

Bit 3 **BATDEN**: Battery Voltage Detection control
Refer to the Battery Voltage Detection section.

Bit 2 Unimplemented, read as "0"

Bit 1~0 **DACVRS1~DACVRS0**: 12-bit D/A Converter Reference Voltage $V_{DACVREF}$ selection
00/01: V_{IREF}
10: AV_{DD}
11: Floating

• **AFEDAC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	AFEDACM1	AFEDACM0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **AFEDACM1~AFEDACM0**: 12-bit D/A Converter mode control
00: DAC Disable, output in a high impedance state
01/11: DAC Enable
10: DAC Disable, output in a ground state

• **AFEDAH & AFEDAL Registers**

Register	AFEDAH								AFEDAL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D11	D10	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	R/W	R/W	—	—	—	—
POR	0	0	0	0	0	0	0	0	0	0	0	0	—	—	—	—

"—": Unimplemented, read as "0"

D11 ~ D0: 12-bit D/A Converter output control bits

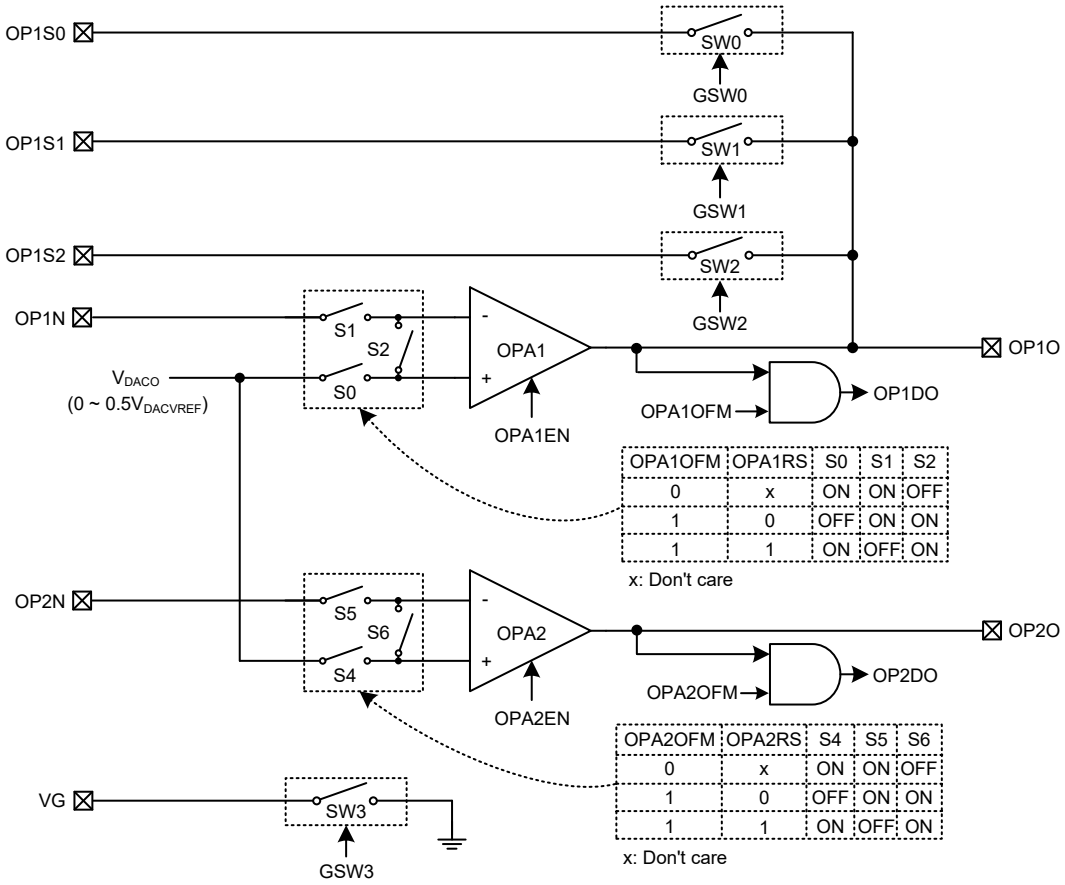
The bit 7~bit 0 in the AFEDAH register combine with the bit 7~bit 4 in the AFEDAL register to form a 12-bit DAC value of 0~4095.

DAC Output Voltage (V_{DACO})=$V_{DACVREF}$×0.5×(DAC value/4096)

Note: It's necessary to firstly write into the AFEDAL register followed by writing into the AFEDAH register.

Operational Amplifiers

This device includes two operational amplifiers for measurement applications. The 12-bit D/A Converter offers a voltage of 0~0.5×$V_{DACVREF}$ to the positive input terminals of the operational amplifiers. The OPAs each provides two operating modes by controlling the switches S0~S2 and S4~S6. By proper setup, the OPA1 and OPA2 output voltage can be connected to the A/D converter internal input channel for measurement.



Operational Amplifier Structure

OPA Register Description

The overall operation of the internal Operational Amplifiers is controlled by a series registers. The OPAnC register together with the OPnDO bit in the IREFC register are used for the OPAn enable/disabled control and input voltage offset calibration settings while the GSC register is for the OPA1 output and VG pin control by setting SW0~SW3 on/off.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OPA1C	OPA1OFM	OPA1RS	OPA1EN	OPA1OF4	OPA1OF3	OPA1OF2	OPA1OF1	OPA1OF0
OPA2C	OPA2OFM	OPA2RS	OPA2EN	OPA2OF4	OPA2OF3	OPA2OF2	OPA2OF1	OPA2OF0
GSC	—	—	—	—	GSW3	GSW2	GSW1	GSW0
IREFC	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0

Operational Amplifier Register List

• **OPA1C Register**

Bit	7	6	5	4	3	2	1	0
Name	OPA1OFM	OPA1RS	OPA1EN	OPA1OF4	OPA1OF3	OPA1OF2	OPA1OF1	OPA1OF0
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 **OPA1OFM**: OPA1 normal operation or input offset voltage calibration mode selection
0: Normal operation
1: Input offset voltage calibration mode

Bit 6 **OPA1RS**: OPA1 input offset voltage calibration reference selection
0: Select the OP1N pin as the reference input
1: Select V_{DACO} as the reference input
Note that the OPA1 input offset voltage calibration can be executed only when OPA1RS=1.

Bit 5 **OPA1EN**: OPA1 enable/disable control
0: Disable
1: Enable

Bit 4~0 **OPA1OF4~OPA1OF0**: OPA1 input offset voltage calibration control bits

• **OPA2C Register**

Bit	7	6	5	4	3	2	1	0
Name	OPA2OFM	OPA2RS	OPA2EN	OPA2OF4	OPA2OF3	OPA2OF2	OPA2OF1	OPA2OF0
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 **OPA2OFM**: OPA2 normal operation or input offset voltage calibration mode selection
0: Normal operation
1: Input offset voltage calibration mode

Bit 6 **OPA2RS**: OPA2 input offset voltage calibration reference selection
0: Select the OP2N pin as the reference input
1: Select V_{DACO} as the reference input
Note that the OPA2 input offset voltage calibration can be executed only when OPA2RS=1.

Bit 5 **OPA2EN**: OPA2 enable/disable control
0: Disable
1: Enable

Bit 4~0 **OPA2OF4~OPA2OF0**: OPA2 input offset voltage calibration control bits

• **GSC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	GSW3	GSW2	GSW1	GSW0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **GSW3**: SW3 (Switch 3) control.
0: Off
1: On
When this bit is set high, the VG pin is connected to ground.

Bit 2 **GSW2**: SW2 (Switch 2) control.
0: Off
1: On
When this bit is set high, the OP1S2 pin is connected to OP1O.

Bit 1 **GSW1**: SW1 (Switch 1) control.
0: Off
1: On
When this bit is set high, the OP1S1 pin is connected to OP1O.

Bit 0 **GSW0**: SW0 (Switch 0) control.
0: Off
1: On
When this bit is set high, the OP1S0 pin is connected to OP1O.

• IREFC Register

Bit	7	6	5	4	3	2	1	0
Name	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0
R/W	R	R	—	R/W	R/W	—	R/W	R/W
POR	0	0	—	0	0	—	0	0

Bit 7 **OP2DO**: OPA2 digital output; positive logic
The OP2DO bit is cleared to zero when the OPA2 is disabled.
When the OPA2OFM bit is set high, the OP2DO bit value indicates the OPA2 output status. Please refer to the Input Offset Calibration procedure.

Bit 6 **OP1DO**: OPA1 digital output; positive logic
The OP1DO bit is cleared to zero when the OPA1 is disabled.
When the OPA1OFM bit is set high, the OP1DO bit value indicates the OPA1 output status. Please refer to the Input Offset Calibration procedure.

Bit 5 Unimplemented, read as "0"

Bit 4 **IREFEN**: Internal Reference Voltage Generator control
Refer to the Internal Reference Voltage Generator section.

Bit 3 **BATDEN**: Battery Voltage Detection control
Refer to the Battery Voltage Detection section.

Bit 2 Unimplemented, read as "0"

Bit 1~0 **DACVRS1~DACVRS0**: 12-bit D/A Converter Reference Voltage $V_{DACVREF}$ selection
Refer to the D/A Converter Registers section.

Input Offset Calibration

The OPAnOFM bit in the OPAnC register is used to select the Operational Amplifier n operating mode, normal operation or input offset calibration mode. The input voltage offset calibration can be executed only when the OPAnRS bit is set to 1. For operational amplifier input offset calibration, the procedures are summarized as the following.

- Step1: Set OPAnOFM=1 and OPAnRS=1, the Operational Amplifier n is now under the offset calibration mode, both of S0 and S2 switches or S4 and S6 switches on. To make sure V_{OS} as minimize as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal mode operation.
- Step2: Set OPAnOF[4:0]=00000, then read the OPnDO bit.
- Step3: Let OPAnOF[4:0]=OPAnOF[4:0] + 1, then read the OPnDO bit.
  If the OPnDO bit state does not change, then repeat Step 3 until the OPnDO bit state changes.
  If the OPnDO bit state changes, record the OPAnOF field value as V_{OS1} and then go to Step 4.
- Step4: Set OPAnOF[4:0]=11111, then read OPnDO bit.
- Step5: Let OPAnOF[4:0]=OPAnOF[4:0] - 1, then read the OPnDO bit.
  If the OPnDO bit state does not change, then repeat Step 5 until the OPnDO bit state changes.
  If the OPnDO bit state changes, record the OPAnOF field value as V_{OS2} and then go to Step 6.

- Step6: Restore the Operational Amplifier n input offset calibration value into the OPAnOF[4:0] bit field. The offset calibration procedure is now finished.
  Where $V_{OS}=(V_{OS1}+V_{OS2})/2$; If $(V_{OS1}+V_{OS2})/2$ is not integral, discard the decimal.

Analog to Digital Converter – ADC

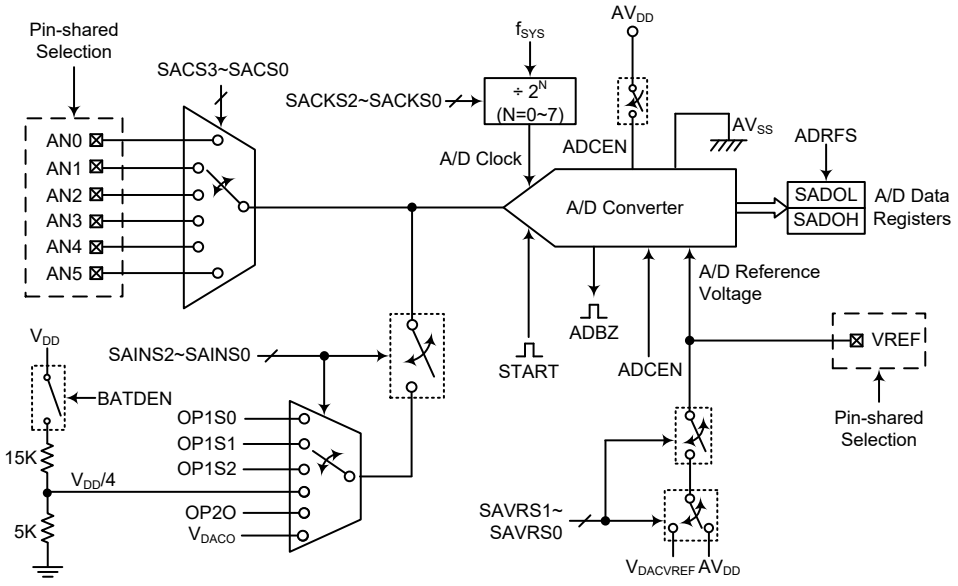
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

This device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the Operational Amplifier output signals (OP1S0, OP1S1, OP1S2 and OP2O), the D/A Converter output signal V_{DACO}, and the divided supply voltage output, $V_{DD}/4$, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 bits together with the SACS3~SACS0 bits. More detailed information about the A/D input signal is described in the "A/D Converter Control Registers" and "A/D Converter Input Signals" sections respectively.

External Input Channels	Internal Signals	Input Signal Selection Bits
6: AN0~AN5	OP1S0, OP1S1, OP1S2, OP2O, V_{DACO}, $V_{DD}/4$	SAINS2~SAINS0, SACS3~SACS0

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



Note: 1. When the OP1Sn (n=0~2) option is selected, the actual analog signal is the OPA1 output, which is implemented using the OP1Sn pin. Details are described in the Operational Amplifier chapter.
2. The BATDEN bit is located in the IREFC register which is described in the Internal Reference Voltage Generator section.

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. The remaining two registers are control registers which setup the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
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
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

A/D Converter Register List

A/D Converter Data Registers – SADOL, SADOH

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

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

A/D Data Registers

A/D Converter Control Registers – SADC0, SADC1

To control the function and operation of the A/D converter, two control registers known as SADC0 and SADC1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external or internal analog signal inputs must be routed to the converter. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input.

• **SADC0 Register**

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

Bit 7 **START**: Start the A/D conversion
0→1→0: Start A/D conversion
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared to 0 again, the A/D converter will initiate a conversion process.

Bit 6 **ADBZ**: A/D converter busy flag
0: No A/D conversion is in progress
1: A/D conversion is in progress
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.

Bit 5 **ADCEN**: A/D converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is cleared to zero, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOH and SADOL will be unchanged.

Bit 4 **ADRFS**: A/D converter data format select
0: A/D converter data format → SADOH=D[11:4]; SADOL=D[3:0]
1: A/D converter data format → SADOH=D[11:8]; SADOL=D[7:0]
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.

Bit 3~0 **SACS3~SACS0**: A/D converter external analog channel input select
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110~1111: Undefined, input floating
These bits are used to select which external analog channel input is to be converted.

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~5 **SAINS2~SAINS0**: A/D converter input signal select
000: External source – External analog channel input
001: Internal source – OPA1 output 0 implemented by the OP1S0 pin
010: Internal source – OPA1 output 1 implemented by the OP1S1 pin
011: Internal source – OPA1 output 2 implemented by the OP1S2 pin
100: Internal source – D/A Converter output, V_{DACO}
101: Internal source – OPA2 output OP2O
110: Internal source – Battery voltage detection output, $V_{DD}/4$
111: External source – External analog channel input

Care must be taken if the SAINS2~SAINS0 bits are set from "001" to "110" to select the internal analog signal to be converted. When the internal analog signal is selected to be converted, the external input pin must never be selected as the A/D input signal by properly setting the SACS3~SACS0 bits with a value from 0110 to 1111. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage select
00/11: External VREF pin
01: Internal A/D converter power, AV_{DD}
10: Internal D/A converter reference voltage, $V_{DACVREF}$
These bits are used to select the A/D converter reference voltage. Care must be taken if the SAVRS1~SAVRS0 bits are set to "01" or "10" to select the internal AV_{DD} or D/A converter reference voltage as the A/D Converter reference voltage source. When the internal A/D converter power or D/A converter reference voltage is selected as the reference voltage, the VREF pin cannot be configured as the reference voltage input by properly configuring the corresponding pin-shared function control bits. Otherwise, the external input voltage on VREF pin will be connected to the selected internal A/D converter reference voltage. This will result in unpredictable situations

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source select
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$
These three bits are used to select the clock source for the A/D converter.

A/D Converter Operation

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 is successfully initiated. When the A/D conversion is complete, the ADBZ will be cleared to 0. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can poll the ADBZ bit in the SADC0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the SACKS2~SACKS0 bits in the SADC1 register. Although the A/D clock source is determined by the system clock f_{SYS} and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken for system clock frequencies. For example, as the system clock operates at a frequency of 8MHz, the SACKS2~SACKS0 bits should not be set to 000, 001 or 111. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D

conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less or larger than the specified A/D Clock Period range.

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

A/D Clock Period Examples

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

A/D Converter Reference Voltage

The A/D converter has its own external reference voltage input pin, VREF. However, the reference voltage can also be supplied from the positive power supply AV_{DD}, the D/A converter reference voltage $V_{DACVREF}$. The choice is made using the SAVRS1 and SAVRS0 bits in the SADC1 register. When the SAVRS bit field is set to “01”, the A/D converter reference voltage will come from the internal power AV_{DD}. When the SAVRS bit field is set to “10”, the A/D converter reference voltage will come from the D/A converter reference voltage $V_{DACVREF}$. When the SAVRS bit field is set to a value except “01” and “10”, the A/D converter reference voltage will come from the external VREF pin. When the VREF pin is selected as the reference voltage supply pin, the relevant pin-shared control bits should first be properly configured to enable the VREF pin function as it is pin-shared with other functions. However, if the internal A/D converter power or the D/A converter reference voltage is selected as the reference voltage, the external VREF pin must not be configured as the reference voltage input function to avoid the internal connection between the VREF pin to the selected internal reference voltage which may result in unpredictable situations. The analog input values must not be allowed to exceed the value of the selected A/D conversion reference voltage.

SAVRS[1:0]	Reference	Description
00, 11	VREF pin	External A/D converter reference input pin VREF
01	AV_{DD}	Internal A/D converter power supply voltage
10	$V_{DACVREF}$	Internal D/A converter reference voltage

A/D Converter Reference Voltage Selection

A/D Converter Input Signals

All of the external A/D analog channel input pins are pin-shared with the I/O pins as well as other functions. The corresponding control bits for each A/D external input pin in the PBS0 and PBS1 registers determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the pin is setup to be as an A/D analog channel input, the original pin functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

There are six internal analog signals derived from the Operational Amplifier output, D/A Converter output signal or Battery Voltage Detection signal, which can be connected to the A/D converter as the analog input signal by configuring the SAINS2~SAINS0 bits. If the external channel input is selected to be converted, the SAINS2~SAINS0 bits should be set to "000" or "111" and the SACS3~SACS0 bits can determine which external channel is selected. Note that if an internal analog signal is selected to be converted, the external input channel determined by the SACS3~SACS0 bits must be switched to a non-existed A/D input channel by properly setting the SACS bit field with a value from 0110 to 1111. Otherwise, the internal analog signal will be connected together with the external channel input. This will result in unpredictable situations.

SAINS[2:0]	SACS[3:0]	Input Signals	Description
000, 111	0000~0101	AN0~AN5	External pin analog input
	0110~1111	—	Non-existed channel, input is floating.
001	0110~1111	OP1S0	Internal OPA1 output 0 implemented by the OP1S0 pin
010	0110~1111	OP1S1	Internal OPA1 output 1 implemented by the OP1S1 pin
011	0110~1111	OP1S2	Internal OPA1 output 1 implemented by the OP1S2 pin
100	0110~1111	V_{DACO}	Internal D/A Converter output
101	0110~1111	OP2O	Internal OPA2 output
110	0110~1111	$V_{DD}/4$	Battery Voltage Detection circuit output

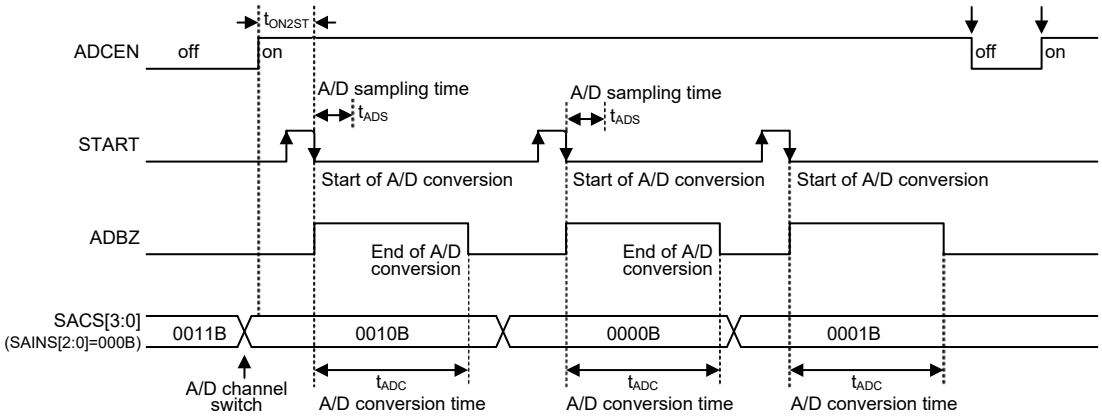
A/D Converter Input Signal Selection

Conversion Rate and Timing Diagram

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

Maximum single A/D conversion rate = 1/(A/D clock period×16)

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



A/D Conversion Timing – External Channel Inputs

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by correctly programming bits SACKS2~SACKS0 in the SADC1 register.
- Step 2
  Enable the A/D by setting the ADCEN bit in the SADC0 register to 1.
- Step 3
  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS2~SAINS0 bits
  Select the external channel input to be converted, go to Step 4.
  Select the internal analog signal to be converted, go to Step 5.
- Step 4
  If the A/D input signal comes from the external channel input selecting by configuring the SAINS bit field, the corresponding pins should be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS bit field. After this step, go to Step 6.
- Step 5
  Before the A/D input signal is selected to come from the internal analog signal by configuring the SAINS bit field, the corresponding external input pin must be switched to a non-existed channel input by setting the SACS3~SACS0 bits with a value from 0110 to 1111. The desired internal analog signal then can be selected by configuring the SAINS bit field. After this step, go to Step 6.
- Step 6
  Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register. If the internal reference voltage is selected, the external reference input pin function must be disabled by properly configuring the corresponding pin-shared control bits.
- Step 7
  Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.
- Step 8
  If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.

- Step 9
  The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10
  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

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

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by clearing ADCEN bit to 0 in the SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Conversion Function

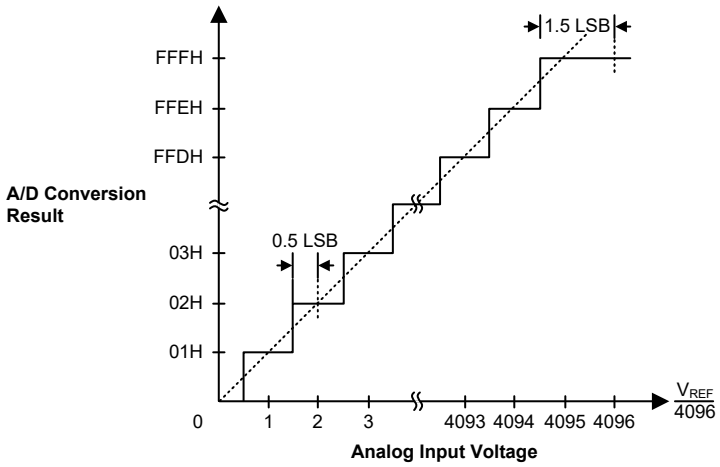
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage, V_{REF}, this gives a single bit analog input value of V_{REF} divided by 4096.

$$1\ LSB=V_{REF} \div 4096$$

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

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

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the V_{REF} level. Note that here the V_{REF} voltage is the actual A/D converter reference voltage determined by the SAVRS field.



Ideal A/D Transfer Function

A/D Conversion Programming Examples

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

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

```
clr  ADE              ; disable ADC interrupt
mov  a,03H
mov  SADC1,a          ; select fSYS/8 as A/D clock
set  ADCEN
mov  a, 30h           ; setup PBS0 to configure pin AN0
mov  PBS0,a
mov  a,20h
mov  SADC0,a          ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr  START            ; high pulse on start bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
polling_EOC:
sz   ADBZ             ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp  polling_EOC      ; continue polling
mov  a,SADOL          ; read low byte conversion result value
mov  SADOL_buffer,a   ; save result to user defined register
mov  a,SADOH          ; read high byte conversion result value
mov  SADOH_buffer,a   ; save result to user defined register
:
:
jmp  start_conversion ; start next A/D conversion
```

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

```
clr  ADE              ; disable ADC interrupt
mov  a,03H
mov  SADC1,a          ; select fSYS/8 as A/D clock
set  ADCEN
mov  a, 30h           ; setup PBS0 to configure pin AN0
mov  PBS0,a
mov  a,20h
mov  SADC0,a          ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr  START            ; high pulse on START bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
clr  ADF              ; clear ADC interrupt request flag
set  ADE              ; enable ADC interrupt
set  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
```

Universal Serial Interface Modules – USIM

The device contains two Universal Serial Interface Modules, USIM0 and USIM1, each of which includes the four-line SPI interface, the two-line I^2C interface and the two-line/single-wire UART interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI, I^2C or UART based hardware such as sensors, Flash or EEPROM memory, etc. Each USIM interface pins are pin-shared with other I/O pins therefore the USIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As all the interface types share the same pins and registers, the choice of whether the USIMn UART, SPI or I^2C type is used is made using the USIMn UART mode selection bit, named UnMD, and the SPI/I^2C operating mode control bits, named SIMn2~SIMn0, in the SIMnC0 register. These pull-high resistors of the USIMn pin-shared I/O are selected using pull-high control registers when the USIMn function is enabled and the corresponding pins are used as USIMn input pins.

SPI Interface

This SPI interface function, which is part of the Universal Serial Interface Modules, should not be confused with the other independent SPI function, which is described in another section of this datasheet.

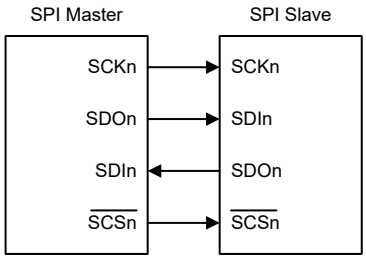
The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the 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 USIMn SPI interface specification can control multiple slave devices from a single master, but the SPI in USIM0 or USIM1 module provides only one $\overline{SCSn}$ 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 USIMn SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDIn, SDOn, SCKn and $\overline{SCSn}$. Pins SDIn and SDOn are the Serial Data Input and Serial Data Output lines, the SCKn pin is the Serial Clock line and $\overline{SCSn}$ is the Slave Select line. As the USIMn SPI interface pins are pin-shared with normal I/O pins and with the USIMn I^2C/UART function pins, the USIMn SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMnC0 and SIMnC2 registers. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls

the clock signal. As an USIMn interface only contains a single $\overline{SCSn}$ pin only one slave device can be utilized. The $\overline{SCSn}$ pin is controlled by software, set CSENn bit to 1 to enable $\overline{SCSn}$ pin function, set CSENn bit to 0 the $\overline{SCSn}$ pin will be floating state.

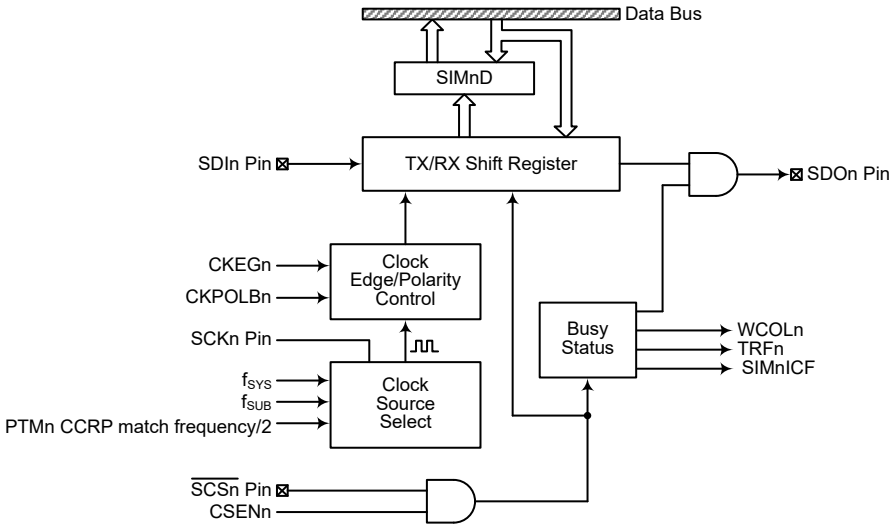


USIMn SPI Master/Slave Connection

The SPI function in the device offers the following features:

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

The status of the USIMn 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 CSENn and SIMnEN.



USIMn SPI Block Diagram (n=0~1)

SPI Registers

There are three internal registers which control the overall operation of the USIMn SPI interface. These are the SIMnD data register and two control registers, SIMnC0 and SIMnC2. Note that the SIMnC2 and SIMnD registers and their POR values are only available when the USIMn SPI mode is selected by properly configuring the UnMD and SIMn2~SIMn0 bits in the SIMnC0 register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMnC0	SIMn2	SIMn1	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
SIMnC2	D7	D6	CKPOLBn	CKEGn	MLSn	CSENn	WCOLn	TRFn
SIMnD	D7	D6	D5	D4	D3	D2	D1	D0

USIMn SPI Register List (n=0~1)

SPI Data Register

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

- **SIMnD 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**: USIMn SPI/I^2C data register bit 7 ~ bit 0

SPI Control Registers

There are also two control registers for the USIMn SPI interface, SIMnC0 and SIMnC2. The SIMnC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMnC2 register is used for other control functions such as LSB/MSB selection, write collision flag etc.

- **SIMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMn2	SIMn1	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SIMn2~SIMn0**: USIMn SPI/I^2C 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 PTMn CCRP match frequency/2
101: SPI slave mode
110: I^2C slave mode
111: Unused mode

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

Bit 4 **UnMD**: USIMn UART mode selection bit
0: USIMn SPI/I²C mode
1: USIMn UART mode
This bit is used to select the USIMn UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIMn2~SIMn0 bits. Note that the UnMD bit must be cleared to zero for USIMn SPI or I²C mode.

Bit 3~2 **SIMnDEB1~SIMnDEB0**: USIMn I²C Debounce Time Selection
These bits are only available when the USIMn is configured to operate in the I²C mode. Refer to the I²C register section.

Bit 1 **SIMnEN**: USIMn SPI/I²C Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the USIMn SPI/I²C interface. When the SIMnEN bit is cleared to zero to disable the USIMn SPI/I²C interface, the SDIn, SDOn, SCKn and $\overline{\text{SCSn}}$, or SDAn and SCLn lines will lose their SPI or I²C function and the USIMn operating current will be reduced to a minimum value. When the bit is high the USIMn SPI/I²C interface is enabled. If the USIMn is configured to operate as an SPI interface via the UnMD and SIMn2~SIMn0 bits, the contents of the USIMn SPI control registers will remain at the previous settings when the SIMnEN bit changes from low to high and should therefore be first initialised by the application program. If the USIMn is configured to operate as an I²C interface via the UnMD and SIMn2~SIMn0 bits and the SIMnEN bit changes from low to high, the contents of the USIMn I²C control bits such as HTXn and TXAKn will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCFn, HAASn, HBBn, SRWn and RXAKn will be set to their default states.

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

- **SIMnC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLBn	CKEGn	MLSn	CSENn	WCOLn	TRFn
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

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

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

Bit 4 **CKEGn**: USIMn SPI active clock edge type selection
CKPOLBn=0
0: SCKn is high base level and data capture at SCKn rising edge
1: SCKn is high base level and data capture at SCKn falling edge

CKPOLBn=1
0: SCKn is low base level and data capture at SCKn falling edge
1: SCKn is low base level and data capture at SCKn rising edge

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

Bit 3 **MLSn**: USIMn 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 **CSENn**: USIMn SPI $\overline{\text{SCSn}}$ pin control
0: Disable
1: Enable

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

Bit 1 **WCOLn**: USIMn SPI write collision flag
0: No collision
1: Collision

The WCOLn flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMnD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared to zero by the application program.

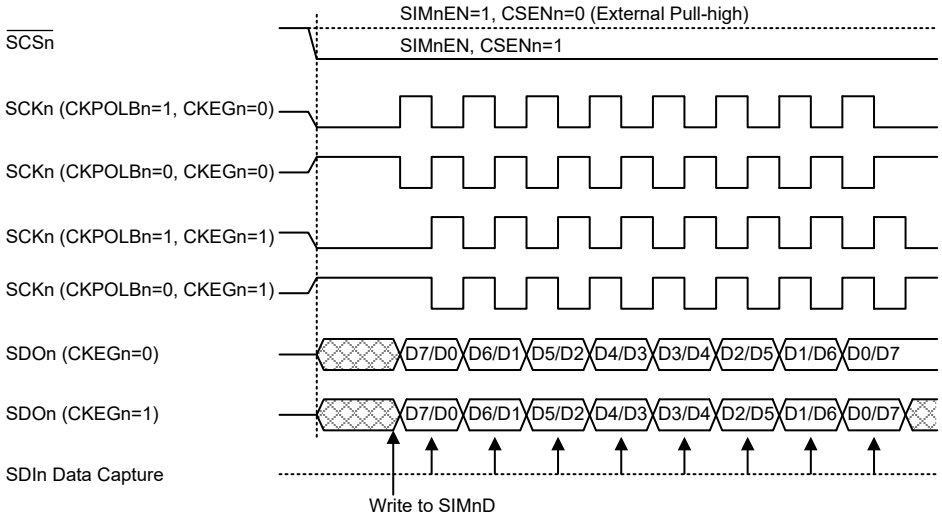
Bit 0 **TRFn**: USIMn SPI Transmit/Receive complete flag
0: USIMn SPI data is being transferred
1: USIMn SPI data transmission is completed

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

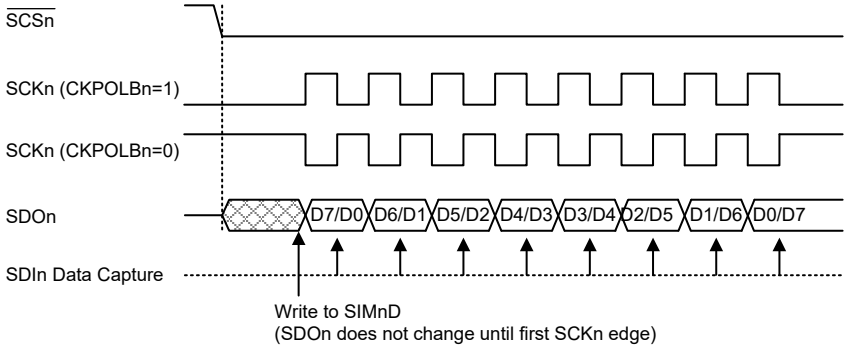
SPI Communication

After the USIMn SPI interface is enabled by setting the SIMnEN bit high, then in the Master Mode, when data is written to the SIMnD register, transmission/reception will begin simultaneously. When the data transfer is completed, the TRFn flag will be set high automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMnD register will be transmitted and any data on the SDIn pin will be shifted into the SIMnD register. The master should output an $\overline{\text{SCSn}}$ 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 SCKn signal depending upon the configurations of the CKPOLBn bit and CKEGn bit. The accompanying timing diagram shows the relationship between the slave data and SCKn signal for various configurations of the CKPOLBn and CKEGn bits.

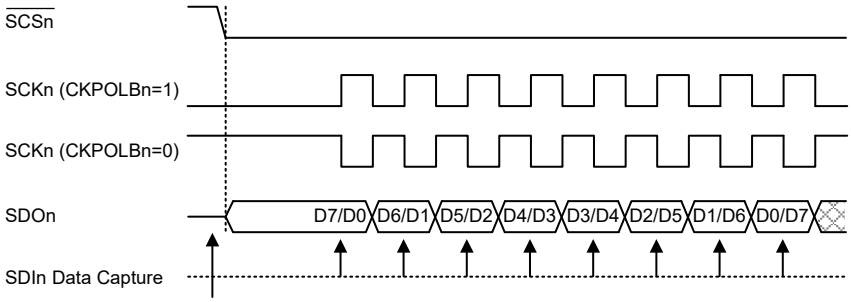
The SPI Master mode will continue to function if the SPI clock is running.



USIMn SPI Master Mode Timing

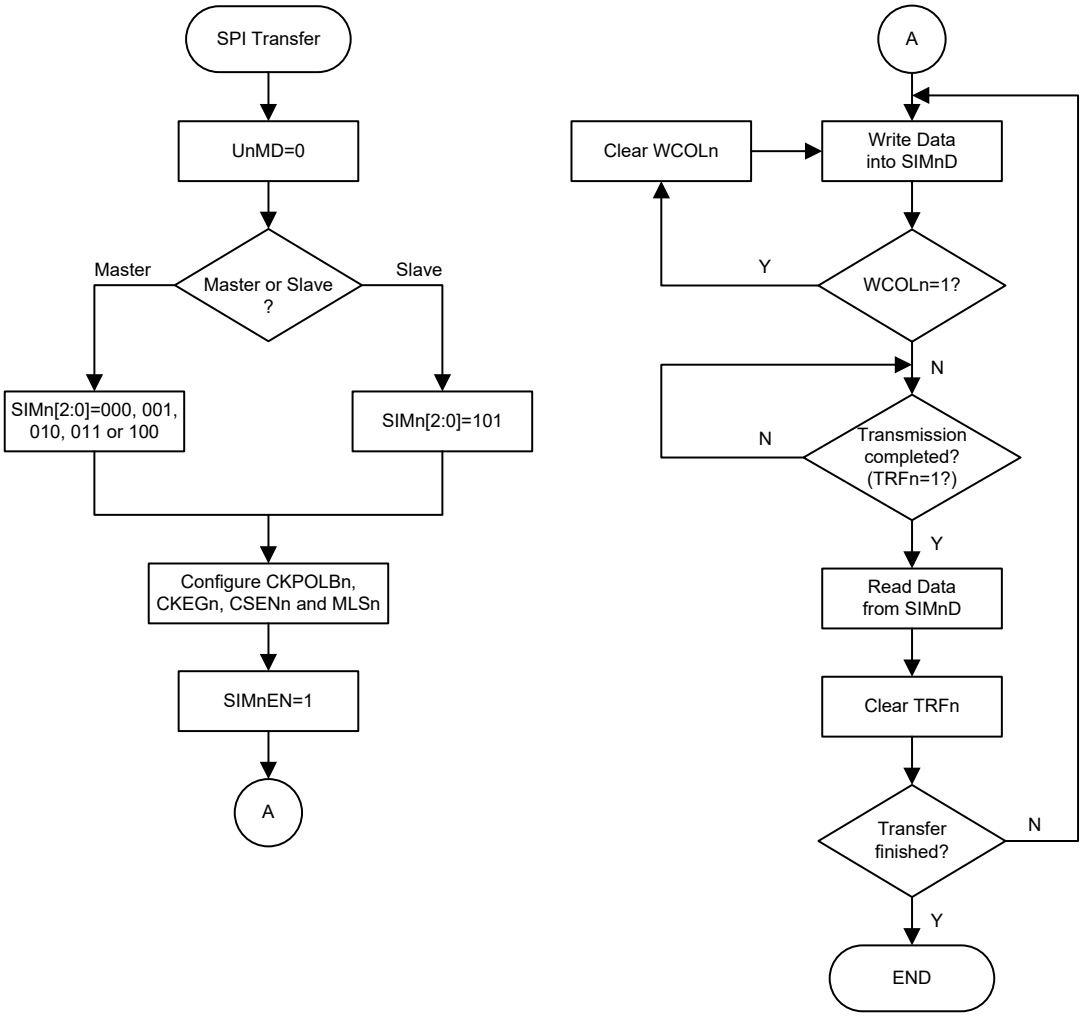


USIMn SPI Slave Mode Timing – CKEGn=0



Note: For SPI slave mode, if SIMnEN=1 and CSENn=0, SPI is always enabled and ignores the SCSn level.

USIMn SPI Slave Mode Timing – CKEGn=1



USIMn SPI Transfer Control Flowchart

SPI Bus Enable/Disable

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

When the USIMn SPI bus is disabled, SCKn, SDIn, SDOn and $\overline{\text{SCSn}}$ can be used as 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 CSENn bit in the SIMnC2 register controls the $\overline{\text{SCSn}}$ pin function of the USIMn SPI interface. Setting this bit high will enable the USIMn SPI interface by allowing the $\overline{\text{SCSn}}$ line to be active, which can then be used to control the USIMn SPI interface. If the CSENn bit is low, the USIMn SPI interface will be disabled and the $\overline{\text{SCSn}}$ line will be in a floating condition and can therefore not be used for control of the USIMn SPI interface. If the CSENn bit and the SIMnEN bit in the SIMnC0 are set high, this will place the SDIn line in a floating condition and the SDOn line high. If in Master

Mode the SCKn line will be either high or low depending upon the clock polarity selection bit CKPOLBn in the SIMnC2 register. If in Slave Mode the SCKn line will be in a floating condition. If the SIMnEN bit is low, then the bus will be disabled and $\overline{\text{SCSn}}$, SDIn, SDOn and SCKn 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 SIMnD 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 USIMn SPI Master mode and clock source using the UnMD and SIMn2~SIMn0 bits in the SIMnC0 control register.
- Step 2
  Setup the CSENn bit and setup the MLSn 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 SIMnEN bit in the SIMnC0 control register to enable the USIMn SPI interface.
- Step 4
  For write operations: write the data to the SIMnD register, which will actually place the data into the TXRX buffer. Then use the SCKn and SDOn lines to output the data. After this, go to step 5.
  For read operations: the data transferred in on the SDIn line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMnD register.
- Step 5
  Check the WCOLn 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 TRFn bit or wait for an USIMn SPI serial bus interrupt.
- Step 7
  Read data from the SIMnD register.
- Step 8
  Clear TRFn.
  Step 9
  Go to step 4.

Slave Mode

- Step 1
  Select the USIMn SPI Slave mode using the UnMD and SIMn2~SIMn0 bits in the SIMnC0 control register
- Step 2
  Setup the CSENn bit and setup the MLSn 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 SIMnEN bit in the SIMnC0 control register to enable the USIMn SPI interface.
- Step 4
  For write operations: write the data to the SIMnD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and $\overline{\text{SCSn}}$ signal. After this, go to step 5.

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

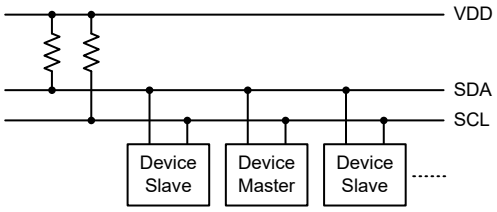
- Step 5
  Check the WCOLn bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the TRFn bit or wait for an USIMn SPI serial bus interrupt.
- Step 7
  Read data from the SIMnD register.
- Step 8
  Clear TRFn.
- Step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

Error Detection

The WCOLn bit in the SIMnC2 register is provided to indicate errors during data transfer. The bit is set by the USIMn 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 SIMnD 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.

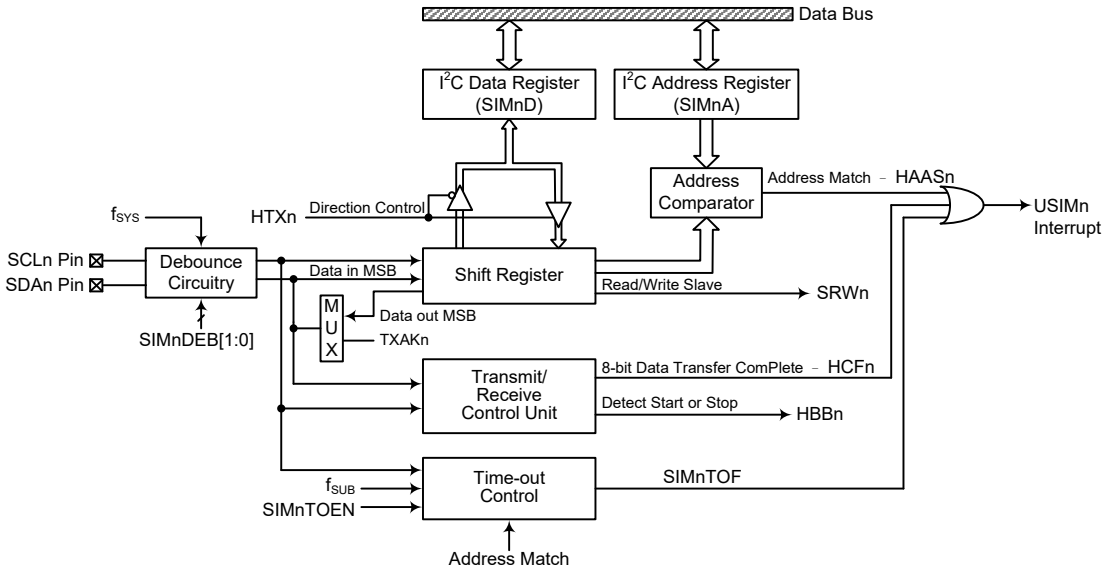


I²C Master Slave Bus Connection

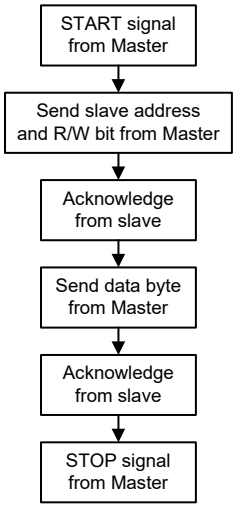
I²C Interface Operation

The I²C serial interface is a two line interface, a serial data line, SDAn, and serial clock line, SCLn. 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 SCLn/SDAn pin is still applicable even if USIMn I²C device is activated and the related internal pull-high register could be controlled by its corresponding pull-high control register.



USIMn I²C Block Diagram (n=0~1)



I²C Interface Operation

The SIMnDEB1 and SIMnDEB0 bits determine the debounce time of the USIMn 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 USIMn I²C bus, SIMnC0, SIMnC1 and SIMnTOC, one address register SIMnA and one data register, SIMnD. Note that the SIMnC1, SIMnD, SIMnA and SIMnTOC registers and their POR values are only available when the USIMn I²C mode is selected by properly configuring the UnMD and SIMn2~SIMn0 bits in the SIMnC0 register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMnC0	SIMn2	SIMn1	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
SIMnC1	HCFn	HAASn	HBBn	HTXn	TXAKn	SRWn	IAMWUn	RXAKn
SIMnD	D7	D6	D5	D4	D3	D2	D1	D0
SIMnA	SIMnA6	SIMnA5	SIMnA4	SIMnA3	SIMnA2	SIMnA1	SIMnA0	D0
SIMnTOC	SIMnTOEN	SIMnTOF	SIMnTOS5	SIMnTOS4	SIMnTOS3	SIMnTOS2	SIMnTOS1	SIMnTOS0

USIMn I²C Register List (n=0~1)

I²C Data Register

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

- **SIMnD 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**: USIMn SPI/I²C data register bit 7 ~ bit 0

I²C Address Register

The SIMnA register is also used by the USIMn SPI interface but has the name SIMnC2. The SIMnA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the SIMnA 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 SIMnA register, the slave device will be selected.

- **SIMnA Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMnA6	SIMnA5	SIMnA4	SIMnA3	SIMnA2	SIMnA1	SIMnA0	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 **SIMnA6~SIMnA0**: USIMn I²C slave address
SIMnA6~SIMnA0 is the 7-bit I²C slave address.

Bit 0 **D0**: Reserved bit, can be read or written by application program

I²C Control Registers

There are three control registers for the USIMn I²C interface, SIMnC0, SIMnC1 and SIMnTOC. The SIMnC0 register is used to control the enable/disable function and to select the I²C slave mode and debounce time. The SIMnC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, SIMnTOC, is used to control the I²C time-out function and is described in the corresponding section.

- **SIMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMn2	SIMn1	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SIMn2~SIMn0**: USIMn SPI/I²C 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 PTMn CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode
When the UnMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIMn function. As well as selecting if the USIMn I²C or SPI function, they are used to control the USIMn SPI Master/Slave selection and the SPI Master clock frequency. The USIMn SPI clock is a function of the system clock but can also be chosen to be sourced from PTMn and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UnMD**: USIMn UART mode selection bit
0: USIMn SPI or I²C mode
1: USIMn UART mode
This bit is used to select the USIMn UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIMn2~SIMn0 bits. Note that the UnMD bit must be cleared to zero for USIMn SPI or I²C mode.

Bit 3~2 **SIMnDEB1~SIMnDEB0**: USIMn I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
10: 4 system clock debounce
11: 4 system clock debounce
These bits are used to select the I²C debounce time when the USIMn is configured as the I²C interface function by setting the UnMD bit to "0" and SIMn2~SIMn0 bits to "110".

Bit 1 **SIMnEN**: USIMn SPI/I²C Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the USIMn SPI/I²C interface. When the SIMnEN bit is cleared to zero to disable the USIMn SPI/I²C interface, the SDIn, SDOn, SCKn and $\overline{\text{SCSn}}$, or SDAn and SCLn lines will lose their SPI or I²C function and the USIMn operating current will be reduced to a minimum value. When the bit is high the USIMn SPI/I²C interface is enabled. If the USIMn is configured to operate as an SPI interface via the UnMD and SIMn2~SIMn0 bits, the contents of the USIMn SPI control registers will remain at the previous settings when the SIMnEN bit changes from low to high and should therefore be first initialised by the application program. If the USIMn is configured to operate as an I²C interface via the UnMD and

SIMn2~SIMn0 bits and the SIMnEN bit changes from low to high, the contents of the USIMn I²C control bits such as HTXn and TXAKn will remain at the previous settings and should therefore be first initialised by the application program while the relevant USIMn I²C flags such as HCFn, HAASn, HBBn, SRWn and RXAKn will be set to their default states.

Bit 0 **SIMnICF**: USIMn SPI Incomplete Flag

This bit is only available when the USIMn is configured to operate in an SPI slave mode. Refer to the SPI register section.

- **SIMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	HCFn	HAASn	HBBn	HTXn	TXAKn	SRWn	IAMWUn	RXAKn
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **HCFn**: USIMn I²C Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer

The HCFn flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Bit 6 **HAASn**: USIMn I²C Bus address match flag
0: Not address match
1: Address match

The HAASn 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 **HBBn**: USIMn I²C Bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy

The HBBn 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 **HTXn**: USIMn I²C slave device is transmitter or receiver selection
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAKn**: USIMn I²C Bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag

The TXAKn 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 TXAKn bit to "0" before further data is received.

Bit 2 **SRWn**: USIMn I²C Slave Read/Write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode

The SRWn flag is the I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C bus. When the transmitted address and slave address is match, that is when the HAASn flag is set high, the slave device will check the SRWn flag to determine whether it should be in transmit mode or receive mode. If the SRWn flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRWn flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

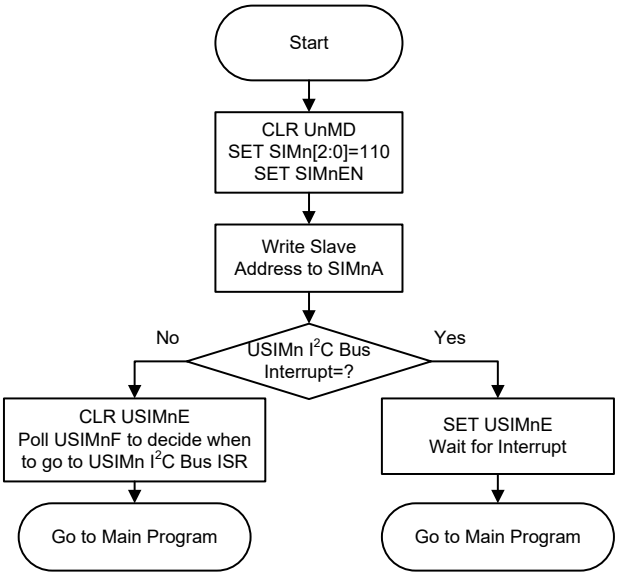
Bit 1 **IAMWUn**: USIMn I²C Address Match Wake-up control
0: Disable
1: Enable
This bit should be set high to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWUn bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake up, then this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

Bit 0 **RXAKn**: USIMn I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave does not receive acknowledge flag
The RXAKn flag is the receiver acknowledge flag. When the RXAKn flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAKn flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAKn flag is "1". When this occurs, the slave transmitter will release the SDAn line to allow the master to send a STOP signal to release the USIMn 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 HAASn bit in the SIMnC1 register will be set and an USIMn interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAASn and SIMnTOF 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 USIMn 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 SRWn 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 UnMD, SIMn2~SIMn0 and SIMnEN bits in the SIMnC0 register to "0", "110" and "1" respectively to enable the USIMn I²C bus.
- Step 2
  Write the slave address of the device to the USIMn I²C bus address register SIMnA.
- Step 3
  Set the USIMnE interrupt enable bit of the interrupt control register to enable the USIMn interrupt.



USIMn 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 HBBn bit will be set. A START condition occurs when a high to low transition on the SDAn line takes place when the SCLn 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 USIMn 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 SRWn bit of the SIMnC1 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 HAASn when the addresses match.

As an USIMn I²C bus interrupt signal can come from three sources, when the program enters the interrupt subroutine, the HAASn and SIMnTOF 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 USIMn 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 SIMnD register, or in the receive mode where it must implement a dummy read from the SIMnD register to release the SCLn line.

I²C Bus Read/Write Signal

The SRWn bit in the SIMnC1 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 SRWn 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 SRWn 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.

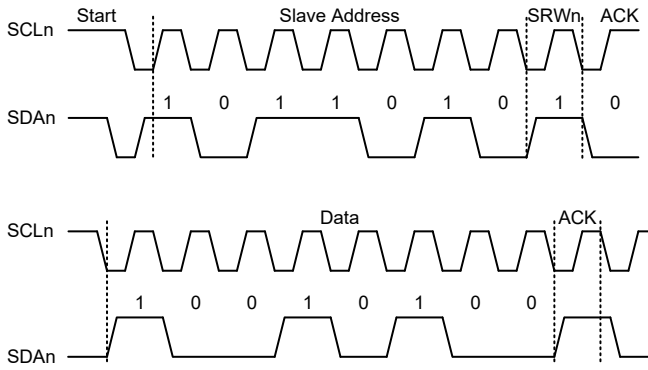
I²C Bus Slave Address Acknowledge Signal

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 HAASn flag is high, the addresses have matched and the slave device must check the SRWn flag to determine if it is to be a transmitter or a receiver. If the SRWn flag is high, the slave device should be setup to be a transmitter so the HTXn bit in the SIMnC1 register should be set to "1". If the SRWn flag is low, then the microcontroller slave device should be setup as a receiver and the HTXn bit in the SIMnC1 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 SDAn line to allow the master to send a STOP signal to release the USIMn I²C Bus. The corresponding data will be stored in the SIMnD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMnD register. If setup as a receiver, the slave device must read the transmitted data from the SIMnD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAKn, on the 9th clock. The slave device, which is setup as a transmitter will check the RXAKn bit in the SIMnC1 register to determine if it is to send another data byte, if not then it will release the SDAn line and wait the receipt of a STOP signal from the master.

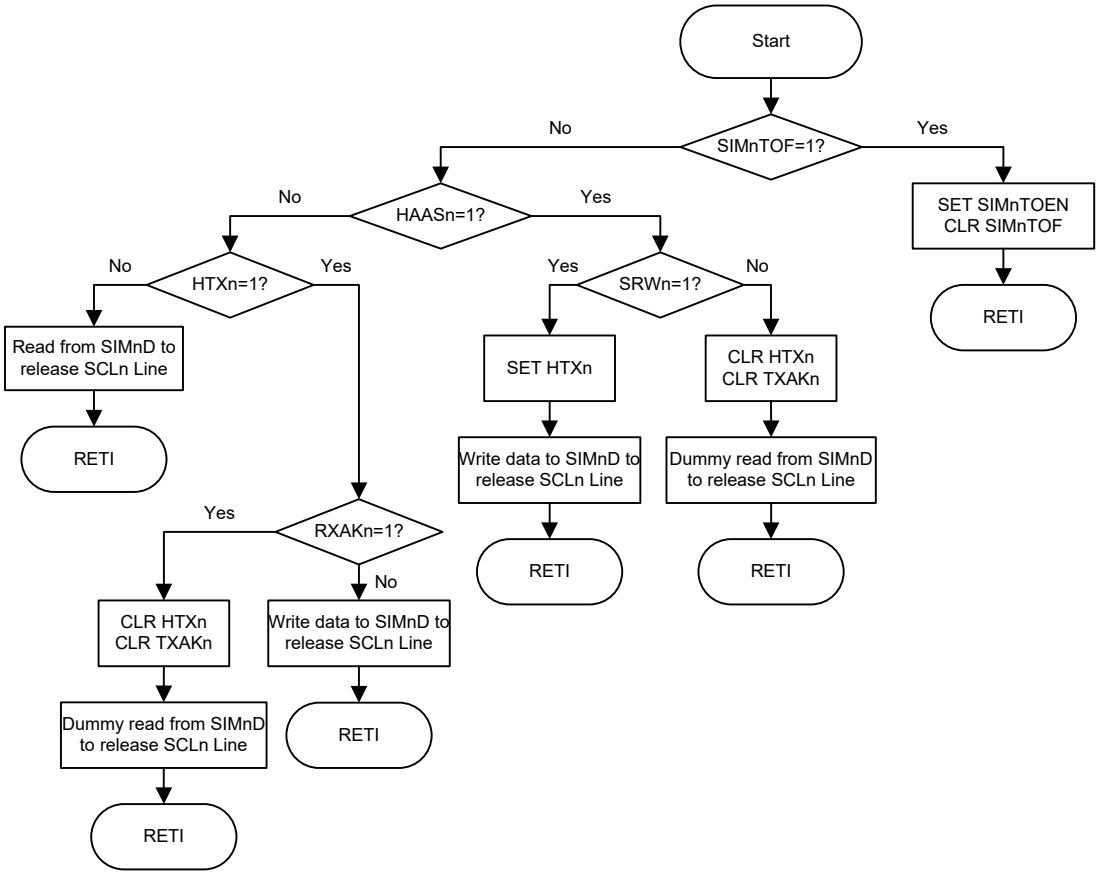


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRWn bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAKn bit for transmitter, TXAKn bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	.....	S	SA	SR	M	D	A	D	A	.....	P

USIMn I²C Communication Timing Diagram

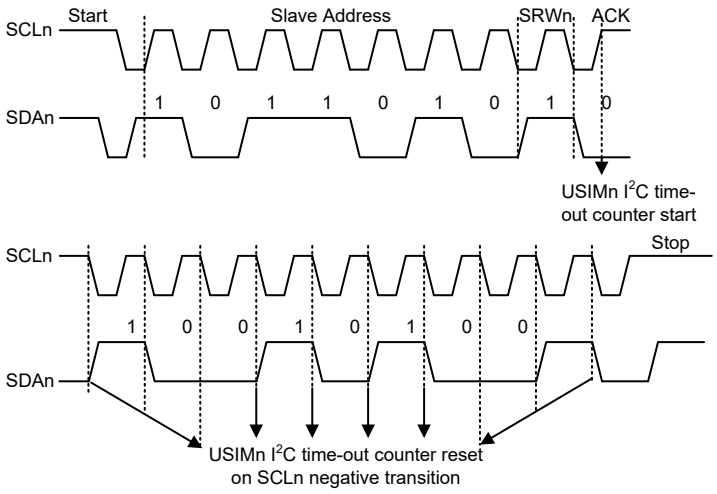
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMnD register, or in the receive mode where it must implement a dummy read from the SIMnD register to release the SCLn line.



USIMn 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 SCLn falling edge. Before the next SCLn falling edge arrives, if the time elapsed is greater than the time-out setup by the SIMnTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.



USIMn I²C Time-out

When an I²C time-out counter overflow occurs, the counter will stop and the SIMnTOEN bit will be cleared to zero and the SIMnTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an USIMn interrupt. 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 USIMn I²C Time-out
SIMnD, SIMnA, SIMnC0	No change
SIMnC1	Reset to POR condition

USIMn I²C Registers after Time-out

The SIMnTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using SIMnTOS bit field in the SIMnTOC register. The time-out time is given by the formula: ((1~64)×32)/f_{SUB}. This gives a time-out period which ranges from about 1ms to 64ms.

- **SIMnTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMnTOEN	SIMnTOF	SIMnTOS5	SIMnTOS4	SIMnTOS3	SIMnTOS2	SIMnTOS1	SIMnTOS0
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 **SIMnTOEN**: USIMn I²C Time-out control
0: Disable
1: Enable

Bit 6 **SIMnTOF**: USIMn 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 to zero by application program.

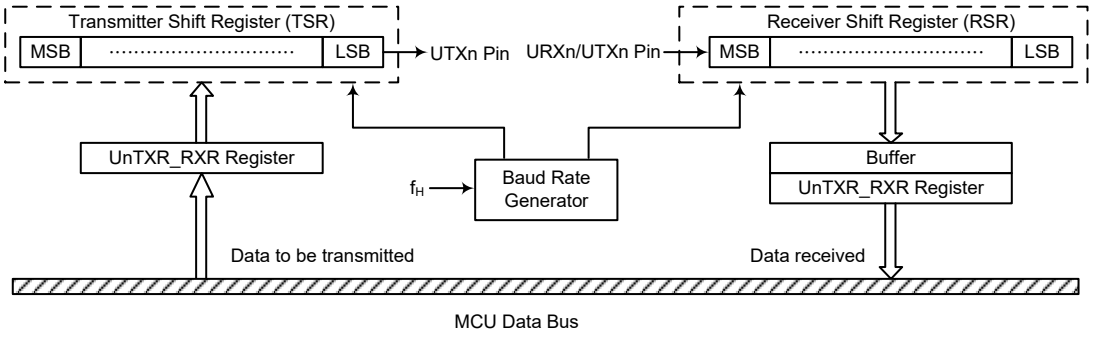
Bit 5~0 **SIMnTOS5~SIMnTOS0**: USIMn I²C Time-out period selection
I²C time-out clock source is f_{SUB}/32.
I²C time-out time is equal to (SIMnTOS[5:0]+1)×(32/f_{SUB}).

UART Interface

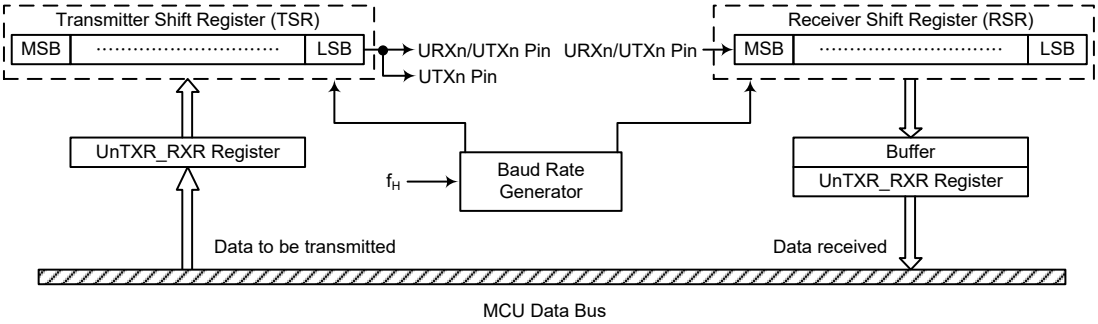
Each USIM contains an integrated full-duplex or half-duplex asynchronous serial communication UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data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

The integrated UART function contains the following features:

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



USIMn UART Data Transfer Block Diagram – UnSWM=0



USIMn UART Data Transfer Block Diagram – UnSWM=1

UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as UTXn pin and URXn/UTXn pin, which are pin-shared with I/O or other pin functions. The UTXn and URXn/UTXn pins are the UART transmitter and receiver pins respectively. The UTXn and URXn/UTXn pin function should first be selected by the corresponding pin-shared function selection register before the USIMn UART function is used. Along with the UnMD bit, the URnEN bit, the UnTXEN or UnRXEN 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 UTXn or URXn/UTXn pin function is disabled by clearing the UnMD, URnEN, UnTXEN or UnRXEN bit, the UTXn or URXn/UTXn pin will be set to a floating state.

UART Single Wire Mode

The UART function also supports a Single Wire Mode communication which is selected using the UnSWM bit in the UnUCR3 register. When the UnSWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single URXn/UTXn pin can be used to transmit and receive data depending upon the corresponding control bits. When the UnRXEN bit is set high, the URXn/UTXn pin is used as a receiver pin. When the UnRXEN bit is cleared to zero and the UnTXEN bit is set high, the URXn/UTXn pin will act as a transmitter pin.

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

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

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

UART Data Transfer Scheme

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

Data to be received by the UART is accepted on the external URXn/UTXn 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 UnTXR_RXR register, where it is buffered and can be manipulated by the application program. Only the UnTXR_RXR register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.

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

UART Status and Control Registers

There are seven control registers associated with the USIMn UART function. The UnMD bit in the SIMnC0 register can be used to select the USIMn UART interface. The UnSWM bit in the UnUCR3 register is used to enable/disable the UART Single Wire Mode. The UnUSR, UnUCR1 and UnUCR2 registers control the overall function of the USIMn UART, while the UnBRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the UnTXR_RXR data register. Note that USIMn UART related registers and their POR values are only available when the UART mode is selected by setting the UnMD bit in the SIMnC0 register to "1".

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMnC0	SIMn2	SIMn1	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
UnUSR	UnPERR	UnNF	UnFERR	UnOERR	UnRIDLE	UnRXIF	UnTIDLE	UnTXIF
UnUCR1	URnEN	UnBNO	UnPREN	UnPRT	UnSTOPS	UnTXBRK	UnRX8	UnTX8
UnUCR2	UnTXEN	UnRXEN	UnBRGH	UnADDEN	UnWAKE	UnRIE	UnTIIE	UnTEIE
UnUCR3	—	—	—	—	—	—	—	UnSWM
UnTXR_RXR	UnTXRX7	UnTXRX6	UnTXRX5	UnTXRX4	UnTXRX3	UnTXRX2	UnTXRX1	UnTXRX0
UnBRG	UnBRG7	UnBRG6	UnBRG5	UnBRG4	UnBRG3	UnBRG2	UnBRG1	UnBRG0

USIMn UART Register List (n=0~1)

- **SIMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMn2	SIMn1	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SIMn2~SIMn0**: USIMn SPI/I²C Operating Mode Control
When the UnMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIMn function. Refer to the SPI or I²C register section for more details.

Bit 4 **UnMD**: USIMn UART mode selection bit
0: USIMn SPI or I²C mode
1: USIMn UART mode
This bit is used to select the USIMn UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIMn2~SIMn0 bits. Note that the UnMD bit must be cleared to zero for USIMn SPI or I²C mode.

Bit 3~2 **SIMnDEB1~SIMnDEB0**: USIMn I²C Debounce Time Selection
Refer to the I²C register section.

Bit 1 **SIMnEN**: USIMn SPI/I²C Enable Control
This bit is only available when the USIMn is configured to operate in an SPI or I²C mode with the UnMD bit low. Refer to the SPI or I²C register section for more details.

Bit 0 **SIMnICF**: USIMn SPI Incomplete Flag
Refer to the SPI register section.

• **UnUSR Register**

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

Bit	7	6	5	4	3	2	1	0
Name	UnPERR	UnNF	UnFERR	UnOERR	UnRIDLE	UnRXIF	UnTIDLE	UnTXIF
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

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

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

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

Bit 4 **UnOERR**: Overrun error flag
0: No overrun error is detected
1: Overrun error is detected
The UnOERR 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 UnTXR_RXR receive data register. The flag is cleared to zero by a software sequence, which is a read to the status register UnUSR followed by an access to the UnTXR_RXR data register.

Bit 3 **UnRIDLE**: Receiver status
0: Data reception is in progress (Data being received)
1: No data reception is in progress (Receiver is idle)
The UnRIDLE 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 UnRIDLE bit is "1" indicating that the UART receiver is idle and the URXn/UTXn pin stays in logic high condition.

Bit 2 **UnRXIF**: Receive UnTXR_RXR data register status
0: UnTXR_RXR data register is empty
1: UnTXR_RXR data register has available data
The UnRXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the UnTXR_RXR read data register is empty. When the flag is "1", it indicates that the UnTXR_RXR read data register contains new data. When the contents of the shift register are transferred to the UnTXR_RXR register, an interrupt is generated if UnRIE=1 in the UnUCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags UnNF, UnFERR, and/or UnPERR are set within the same clock cycle. The UnRXIF flag will eventually be cleared to zero when the UnUSR register is read with UnRXIF set, followed by a read from the UnTXR_RXR register, and if the UnTXR_RXR register has no more new data available.

Bit 1 **UnTIDLE**: Transmission idle
0: Data transmission is in progress (Data being transmitted)
1: No data transmission is in progress (Transmitter is idle)
The UnTIDLE 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 UnTXIF flag is "1" and when there is no transmit data or break character being transmitted. When UnTIDLE is equal to "1", the UTXn pin becomes idle with the pin state in logic high condition. The UnTIDLE flag is cleared to zero by reading the UnUSR register with UnTIDLE set and then writing to the UnTXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 **UnTXIF**: Transmit UnTXR_RXR data register status
0: Character is not transferred to the transmit shift register
1: Character has transferred to the transmit shift register (UnTXR_RXR data register is empty)
The UnTXIF 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 UnTXR_RXR data register. The UnTXIF flag is cleared to zero by reading the UART status register (UnUSR) with UnTXIF set and then writing to the UnTXR_RXR data register. Note that when the UnTXEN bit is set, the UnTXIF flag bit will also be set since the transmit data register is not yet full.

• UnUCR1 Register

The UnUCR1 register together with the UnUCR2 and UnUCR3 registers are the three UART control registers that are used to set the various options for the USIMn UART function, such as overall on/off control, parity control, data transfer bit length, single wire mode communication etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	URnEN	UnBNO	UnPREN	UnPRT	UnSTOPS	UnTXBRK	UnRX8	UnTX8
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 **URnEN**: USIMn UART function enable control
0: Disable UART. UTXn and URXn/UTXn pins are in a floating state
1: Enable UART. UTXn and URXn/UTXn pins function as UART pins
The URnEN bit is the USIMn UART enable bit. When this bit is equal to "0", the UART will be disabled and the URXn/UTXn pin as well as the UTXn pin will be set in a floating state. When the bit is equal to "1", the UART will be enabled if the UnMD bit is set and the UTXn and URXn/UTXn pins will function as defined by the UnSWM mode selection bit together with the UnTXEN and UnRXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the UnTXEN, UnRXEN, UnTXBRK, UnRXIF, UnOERR, UnFERR, UnPERR and UnNF bits will be cleared to zero, while the UnTIDLE, UnTXIF and UnRIDLE bits will be set high. Other control bits in UnUCR1, UnUCR2, UnUCR3 and UnBRG registers will remain unaffected. If the UART is active and the URnEN bit is cleared to zero, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6 **UnBNO**: 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 UnRX8 and UnTX8 will be used to store the 9th bit of the received and transmitted data respectively.

Bit 5 **UnPREN**: Parity function enable control
0: Parity function is disabled
1: Parity function is enabled

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

Bit 4 **UnPRT**: Parity type selection bit
0: Even parity for parity generator
1: Odd parity for parity generator

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

Bit 3 **UnSTOPS**: Number of Stop bits selection for transmitter
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 transmitter. When this bit is equal to "1", two stop bits are used. If this bit is equal to "0", then only one stop bit is used.

Bit 2 **UnTXBRK**: Transmit break character
0: No break character is transmitted
1: Break characters transmit

The UnTXBRK bit is the Transmit Break Character bit. When this bit is "0", there are no break characters and the UTXn 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 UnTXBRK bit is reset.

Bit 1 **UnRX8**: 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 UnRX8. The UnBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **UnTX8**: 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 UnTX8. The UnBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UnUCR2 Register**

The UnUCR2 register is the second of the USIMn UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UART Transmitter and Receiver as well as enabling the various USIMn UART mode interrupt sources. The register also serves to control the baud rate speed, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UnTXEN	UnRXEN	UnBRGH	UnADDEN	UnWAKE	UnRIE	UnTIIE	UnTEIE
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 **UnTXEN**: USIMn UART Transmitter enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled
The bit named UnTXEN 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 UTXn pin will be set in a floating state.
If the UnTXEN bit is equal to "1" and the UnMD and URnEN bit are also equal to "1", the transmitter will be enabled and the UTXn pin will be controlled by the UART. Clearing the UnTXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the UTXn pin will be set in a floating state.

Bit 6 **UnRXEN**: USIMn UART Receiver enabled control
0: UART receiver is disabled
1: UART receiver is enabled
The bit named UnRXEN 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 URXn/UTXn pin will be set in a floating state. If the UnRXEN bit is equal to "1" and the UnMD and URnEN bit are also equal to "1", the receiver will be enabled and the URXn/UTXn pin will be controlled by the UART. Clearing the UnRXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the URXn/UTXn pin will be set in a floating state.

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

Bit 4 **UnADDEN**: Address detect function enable control
0: Address detect function is disabled
1: Address detect function is enabled
The bit named UnADDEN 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 UnTXRX7 if UnBNO=0 or the 9th bit, which corresponds to UnRX8 if UnBNO=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 UnBNO. 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 **UnWAKE**: URXn/UTXn pin wake-up USIMn UART function enable control
0: URXn/UTXn pin wake-up UART function is disabled
1: URXn/UTXn pin wake-up UART function is enabled
This bit is used to control the wake-up USIMn UART function when a falling edge on the URXn/UTXn pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no URXn/UTXn pin wake-up UART function if the UART clock (f_H) exists. If the UnWAKE bit is set high as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the URXn/UTXn pin occurs. When this request happens and the corresponding interrupt is enabled, an URXn/UTXn pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the URXn/UTXn pin when the UnWAKE bit is cleared to zero.

Bit 2 **UnRIE**: 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 UnOERR or receive data available flag UnRXIF is set, the USIMn interrupt request flag USIMnF will be set. If this bit is equal to "0", the USIMn interrupt request flag USIMnF will not be influenced by the condition of the UnOERR or UnRXIF flags.

Bit 1 **UnTIIE**: 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 UnTIDLE is set, due to a transmitter idle condition, the USIMn interrupt request flag USIMnF will be set. If this bit is equal to "0", the USIMn interrupt request flag USIMnF will not be influenced by the condition of the UnTIDLE flag.

Bit 0 **UnTEIE**: 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 UnTXIF is set, due to a transmitter empty condition, the USIMn interrupt request flag USIMnF will be set. If this bit is equal to "0", the USIMn interrupt request flag USIMnF will not be influenced by the condition of the UnTXIF flag.

• UnUCR3 Register

The UnUCR3 register is used to enable the USIMn UART Single Wire Mode communication. As the name suggests in the single wire mode the UART communication can be implemented in one single line, URXn/UTXn, together with the control of the UnRXEN and UnTXEN bits in the UnUCR2 register.

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **UnSWM**: Single Wire Mode enable control
0: Disable, the URXn/UTXn pin is used as UART receiver function only
1: Enable, the URXn/UTXn pin can be used as UART receiver or transmitter function controlled by the UnRXEN and UnTXEN bits
Note that when the Single Wire Mode is enabled, if both the UnRXEN and UnTXEN bits are high, the URXn/UTXn pin will just be used as UART receiver input.

• **UnTXR_RXR Register**

The UnTXR_RXR register is the data register which is used to store the data to be transmitted on the UTXn pin or being received from the URXn/UTXn pin.

Bit	7	6	5	4	3	2	1	0
Name	UnTXRX7	UnTXRX6	UnTXRX5	UnTXRX4	UnTXRX3	UnTXRX2	UnTXRX1	UnTXRX0
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 **UnTXRX7~UnTXRX0**: UART Transmit/Receive Data bit 7 ~ bit 0

• **UnBRG Register**

Bit	7	6	5	4	3	2	1	0
Name	UnBRG7	UnBRG6	UnBRG5	UnBRG4	UnBRG3	UnBRG2	UnBRG1	UnBRG0
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 **UnBRG7~ UnBRG0**: Baud Rate values

By programming the UnBRGH bit in UnUCR2 register which allows selection of the related formula described above and programming the required value in the UnBRG register, the required baud rate can be setup.

Note: Baud rate=f_H/[64×(N+1)] if UnBRGH=0.

Baud rate=f_H/[16×(N+1)] if UnBRGH=1.

Baud Rate Generator

To setup the speed of the serial data communication, the USIMn UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the baud rate register UnBRG and the second is the value of the UnBRGH bit in the UnUCR2 control register. The UnBRGH bit decides if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value N in the UnBRG register which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the UnBRG register and has a range of between 0 and 255.

UnBRGH Bit	0	1
USIMn UART Baud Rate (BR)	f_H/[64×(N+1)]	f_H/[16×(N+1)]

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

Calculating the Baud Rate and Error Values

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

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

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

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

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

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

UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding UnBNO, UnPRT, UnPREN, and UnSTOPS bits in the UnUCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

The basic on/off function of the internal USIMn UART function is controlled using the URnEN bit in the UnUCR1 register. When the USIMn UART mode is selected by setting the UnMD bit in the SIMnC0 register to "1", if the URnEN, UnTXEN and UnRXEN bits are set, then these two UART pins will act as normal UTXn output pin and URXn/UTXn input pin respectively. If no data is being transmitted on the UTXn pin, then it will default to a logic high value.

Clearing the URnEN bit will disable the UTXn and URXn/UTXn pin and allow these pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits UnTXEN, UnRXEN, UnTXBRK, UnRXIF, UnOERR, UnFERR, UnPERR and UnNF being cleared while bits UnTIDLE, UnTXIF and UnRIDLE will be set. The remaining control bits in the UnUCR1, UnUCR2, UnUCR3 and UnBRG registers will remain unaffected. If the URnEN bit in the UnUCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

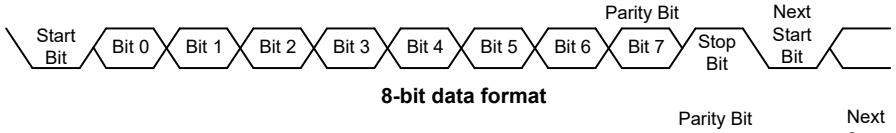
Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UnUCR1 register. The UnBNO bit controls the number of data bits which can be set to either 8 or 9, the UnPRT bit controls the choice of odd or even parity, the UnPREN bit controls the parity on/off function and the UnSTOPS bit decides whether one or two stop bits are to be used. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only used for the transmitter. There is only one stop bit for the receiver.

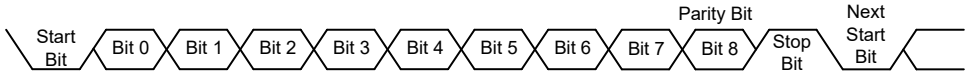
Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
Example of 9-bit Data Formats				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

Transmitter Receiver Data Format

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



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 UnBNO bit in the UnUCR1 register. When UnBNO 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 UnTX8 bit in the UnUCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the UnTXR_RXR register. The data to be transmitted is loaded into this UnTXR_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the UnTXR_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the UnTXEN bit is set, but the data will not be transmitted until the UnTXR_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the UnTXR_RXR register, after which the UnTXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the UnTXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the UnTXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The UTXn output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

Transmitting Data

When the UART is transmitting data, the data is shifted on the UTXn pin from the shift register, with the least significant bit first. In the transmit mode, the UnTXR_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the UnTX8 bit in the UnUCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the UnBNO, UnPRT, UnPREN and UnSTOPS bits to define the required word length, parity type and number of stop bits.
- Setup the UnBRG register to select the desired baud rate.
- Set the UnTXEN bit ensure that the UTXn pin is used as a UART transmitter pin.
- Access the UnUSR register and write the data that is to be transmitted into the UnTXR_RXR register. Note that this step will clear the UnTXIF bit.

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

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

1. A UnUSR register access
2. A UnTXR_RXR register write execution

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

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

1. A UnUSR register access
2. A UnTXR_RXR register write execution

Note that both the UnTXIF and UnTIDLE bits are cleared by the same software sequence.

Transmitting Break

If the UnTXBRK bit is set high and the state keeps for a time greater than [(UnBRG+1)×t_H], then the break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13×N '0' bits and stop bits, where N=1, 2, etc. If a break character is to be transmitted then the UnTXBRK 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 UnTXBRK 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 UnTXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out one or two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the UnBNO bit is set, the word length will be set to 9 bits with the MSB being stored in the UnRX8 bit of the UnUCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the URXn/UTXn pin input 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 URXn/UTXn pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on

the external URXn/UTXn pin input is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the URXn/UTXn pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

Receiving Data

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

- Make the correct selection of UnBNO, UnPRT and UnPREN bits to define the word length, parity type.
- Setup the UnBRG register to select the desired baud rate.
- Set the UnRXEN bit to ensure that the URXn/UTXn pin is used as a UART receiver pin.

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

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

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

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

1. A UnUSR register access
2. A UnTXR_RXR register read execution

Receiving Break

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

- The framing error flag, UnFERR, will be set.
- The receive data register, UnTXR_RXR, will be cleared.
- The UnOERR, UnNF, UnPERR, UnRIDLE or UnRXIF 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 UnUSR register, otherwise known as the UnRIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the UnRIDLE flag will have a high value, which indicates the receiver is in an idle condition.

Receiver Interrupt

The read only receive interrupt flag UnRXIF in the UnUSR register is set by an edge generated by the receiver. An interrupt is generated if UnRIE=1, when a word is transferred from the Receive Shift Register, RSR, to the Receive Data Register, UnTXR_RXR. An overrun error can also generate an interrupt if UnRIE=1.

Managing Receiver Errors

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

Overrun Error – UnOERR

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

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

- The UnOERR flag in the UnUSR register will be set.
- The UnTXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the UnRIE bit is set.

The UnOERR flag can be cleared by an access to the UnUSR register followed by a read to the UnTXR_RXR register.

Noise Error – UnNF

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, UnNF, in the UnUSR register will be set on the rising edge of the UnRXIF bit.
- Data will be transferred from the Shift register to the UnTXR_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the UnRXIF bit which itself generates an interrupt.

Note that the UnNF flag is reset by a UnUSR register read operation followed by a UnTXR_RXR register read operation.

Framing Error – UnFERR

The read only framing error flag, UnFERR, in the UnUSR 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 UnFERR flag will be set. The UnFERR flag and the received data will be recorded in the UnUSR and UnTXR_RXR registers respectively, and the flag is cleared in any reset.

Parity Error – UnPERR

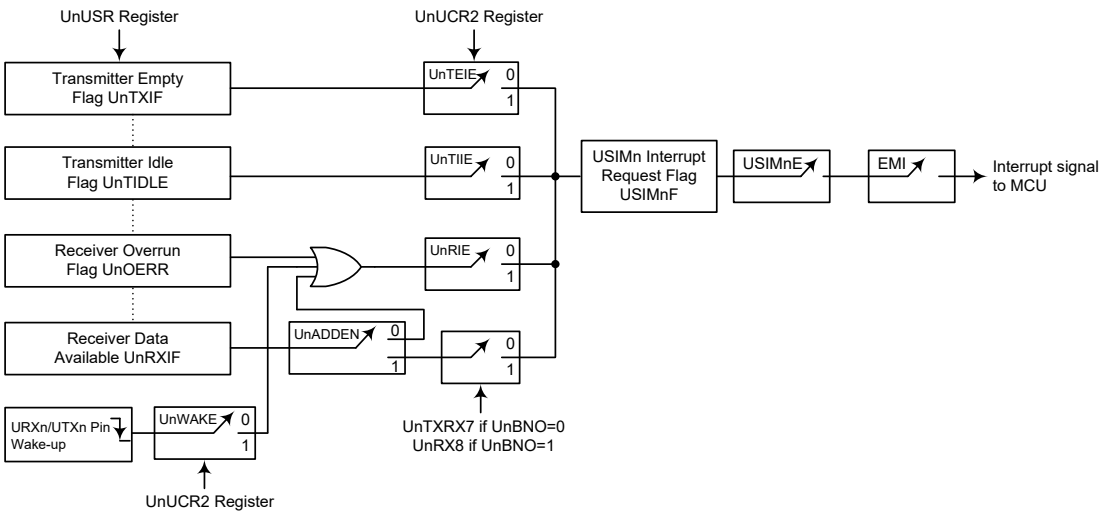
The read only parity error flag, UnPERR, in the UnUSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, UnPREN=1, and if the parity type, odd or even is selected. The read only UnPERR flag and the received data will be recorded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

UART Interrupt Structure

Several individual UART conditions can trigger an USIMn interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an URXn/UTXn pin wake-up. When any of these conditions are created, if the global interrupt enable bit and the USIMn 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 UnUSR register flags which will generate an USIMn interrupt if its associated interrupt enable control bit in the UnUCR2 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 USIMn UART mode interrupt sources.

The address detect condition, which is also an USIMn UART mode interrupt source, does not have an associated flag, but will generate an USIMn interrupt when an address detect condition occurs if its function is enabled by setting the UnADDEN bit in the UnUCR2 register. An URXn/UTXn pin wake-up, which is also an USIMn UART mode interrupt source, does not have an associated flag, but will generate an USIMn interrupt if the UART clock (f_H) source is switched off and the UnWAKE and UnRIE bits in the UnUCR2 register are set when a falling edge on the URXn/UTXn pin occurs. Note that in the event of an URXn/UTXn wake-up interrupt occurring, there will be a certain period of delay, commonly known as the System Start-up Time, for the oscillator to restart and stabilize before the system resumes normal operation.

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



USIMn UART Interrupt Structure

Address Detect Mode

Setting the Address Detect Mode bit, UnADDEN, in the UnUCR2 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 UnRXIF flag. If the UnADDEN 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 USIMnE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if UnBNO=1 or the 8th bit if UnBNO=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 UnADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the UnRXIF 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 UnPREN to zero.

UnADDEN	9th bit if UnBNO=1 8th bit if UnBNO=0	USIMn Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

UnADDEN Bit Function

UART Power Down and Wake-up

When the UART clock (f_H) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock (f_H) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the IDLE or SLEEP mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the UnUSR, UnUCR1, UnUCR2, UnUCR3, UnTXR_RXR, as well as the UnBRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver URXn/UTXn pin wake-up function, which is enabled or disabled by the UnWAKE bit in the UnUCR2 register. If this bit, along with the USIMn UART mode selection bit, UnMD, the UART enable bit, URnEN, the receiver enable bit, UnRXEN and the receiver interrupt bit, UnRIE, are all set when the UART clock (f_H) is off, then a falling edge on the URXn/UTXn pin will trigger an URXn/UTXn pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the URXn/UTXn pin will be ignored.

For a USIMn UART wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the USIMn interrupt enable bit, USIMnE, must be set. If the EMI and USIMnE 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 USIMn interrupt will not be generated until after this time has elapsed.

Serial Peripheral Interface – SPI

The device contains an independent SPI function. It is important not to confuse this independent SPI function with the additional two contained within the combined USIM0~USIM1 functions, which are described in another section of this datasheet.

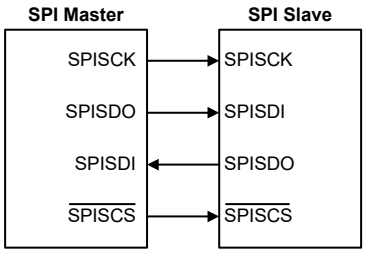
The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the 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, however the device provides only one $\overline{\text{SPISCS}}$ pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pins to select the slave devices.

SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SPISDI, SPISDO, SPISCK and $\overline{\text{SPISCS}}$. Pins SPISDI and SPISDO are the Serial Data Input and Serial Data Output lines, the SPISCK pin is the Serial Clock line and $\overline{\text{SPISCS}}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins, the SPI interface must first be enabled by configuring the corresponding selection bits in the pin-shared function selection registers. The SPI can be disabled or enabled using the SPIEN bit in the SPIC0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{\text{SPISCS}}$ pin only one slave device can be utilized. The pull-high resistors of the SPI pin-shared I/O are selected using pull-high control registers when the SPI function is enabled and the corresponding pins are used as SPI input pins.

The $\overline{\text{SPISCS}}$ pin is controlled by software, set SPICSEN bit to 1 to enable the $\overline{\text{SPISCS}}$ pin function, and clear SPICSEN bit to 0, the $\overline{\text{SPISCS}}$ pin will be floating state.

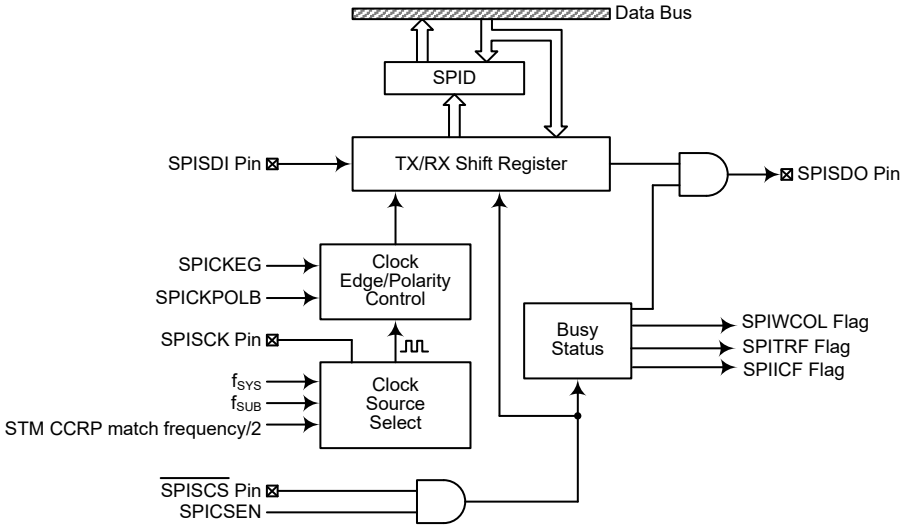


SPI Master/Slave Connection

The SPI function in the device offers the following features:

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

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



SPI Block Diagram

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SPID data register and two registers, SPIC0 and SPIC1.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SPIC0	SPIM2	SPIM1	SPIM0	—	—	—	SPIEN	SPIICF
SPIC1	—	—	SPICKPOLB	SPICKEG	SPIMLS	SPICSEN	SPIWCOL	SPITRF
SPID	D7	D6	D5	D4	D3	D2	D1	D0

SPI Register List

SPI Data Register

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

SPID 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**: SPI data register bit 7 ~ bit 0

SPI Control Registers

There are also two control registers for the SPI interface, SPIC0 and SPIC1. The SPIC0 register is used to control the enable/disable function and to select the data transmission clock frequency. The SPIC1 register is used for other control functions such as LSB/MSB selection, write collision flag etc.

• SPIC0 Register

Bit	7	6	5	4	3	2	1	0
Name	SPIM2	SPIM1	SPIM0	—	—	—	SPIEN	SPIICF
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	1	1	1	—	—	—	0	0

Bit 7~5 **SPIM2~SPIM0**: SPI 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 STM CCRP match frequency/2
101: SPI slave mode
110/111: SPI disable
These bits 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 STM and f_{SUB}. If the SPI slave mode is selected then the clock will be supplied by an external Master device.

Bit 4~2 Unimplemented, read as "0"

Bit 1 **SPIEN**: SPI enable control
0: Disable
1: Enable
The bit is the overall on/off control for the SPI interface. When the SPIEN bit is cleared to zero to disable the SPI interface, the SPISDI, SPISDO, SPISCK and $\overline{\text{SPISCS}}$ lines will lose their SPI function and the SPI operating current will be reduced to a minimum value. When the bit is high the SPI interface is enabled.

Bit 0 **SPIICF**: SPI incomplete flag
0: SPI incomplete condition is not occurred
1: SPI incomplete condition is occured
This bit is only available when the SPI is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SPIEN and SPICSEN bits both being set high but the $\overline{\text{SPISCS}}$ line is pulled high by the external master device before the

SPI data transfer is completely finished, the SPIICF bit will be set high together with the SPITRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the SPITRF bit will not be set high if the SPIICF bit is set high by software application program.

- **SPIC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SPICKPOLB	SPICKEG	SPIMLS	SPICSEN	SPIWCOL	SPITRF
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 **SPICKPOLB**: SPI clock line base condition selection
0: The SPISCK line will be high when the clock is inactive
1: The SPISCK line will be low when the clock is inactive
The SPICKPOLB bit determines the base condition of the clock line, if the bit is high, then the SPISCK line will be low when the clock is inactive. When the SPICKPOLB bit is low, then the SPISCK line will be high when the clock is inactive.

Bit 4 **SPICKEG**: SPI SPISCK clock active edge type selection
SPICKPOLB=0
0: SPISCK has high base level with data capture on SPISCK rising edge
1: SPISCK has high base level with data capture on SPISCK falling edge
SPICKPOLB=1
0: SPISCK has low base level with data capture on SPISCK falling edge
1: SPISCK has low base level with data capture on SPISCK rising edge
The SPICKEG and SPICKPOLB 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 a data transfer is executed otherwise an erroneous clock edge may be generated. The SPICKPOLB bit determines the base condition of the clock line, if the bit is high, then the SPISCK line will be low when the clock is inactive. When the SPICKPOLB bit is low, then the SPISCK line will be high when the clock is inactive. The SPICKEG bit determines active clock edge type which depends upon the condition of the SPICKPOLB bit.

Bit 3 **SPIMLS**: 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 **SPICSEN**: SPI $\overline{\text{SPISCS}}$ pin control
0: Disable
1: Enable
The SPICSEN bit is used as an enable/disable for the $\overline{\text{SPISCS}}$ pin. If this bit is low, then the $\overline{\text{SPISCS}}$ pin will be disabled and placed into a floating condition. If the bit is high the $\overline{\text{SPISCS}}$ pin will be enabled and used as a select pin.

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

Bit 0 **SPITRF**: SPI Transmit/Receive complete flag
0: SPI data is being transferred
1: SPI data transmission is completed

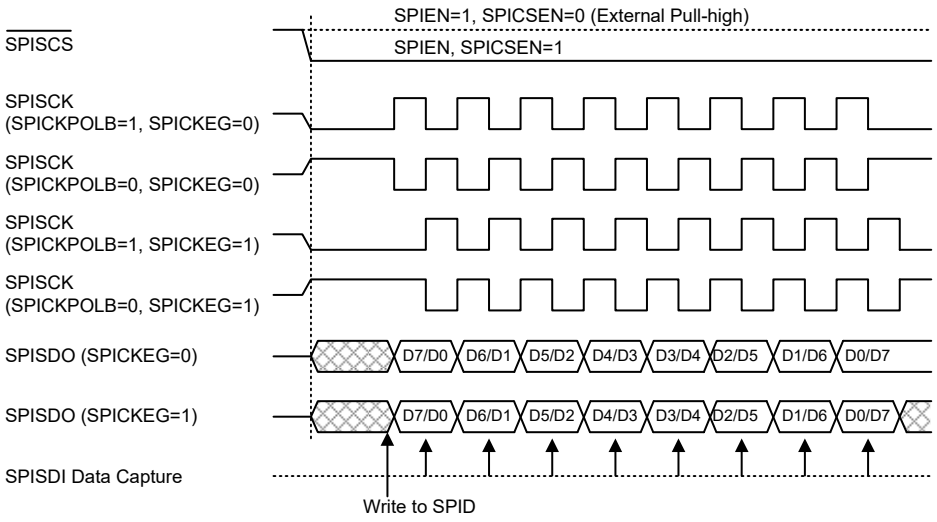
The SPITRF bit is the Transmit/Receive Complete flag and is set "1" automatically when an SPI data transmission is completed, but must set to zero by the application program. It can be used to generate an interrupt.

SPI Communication

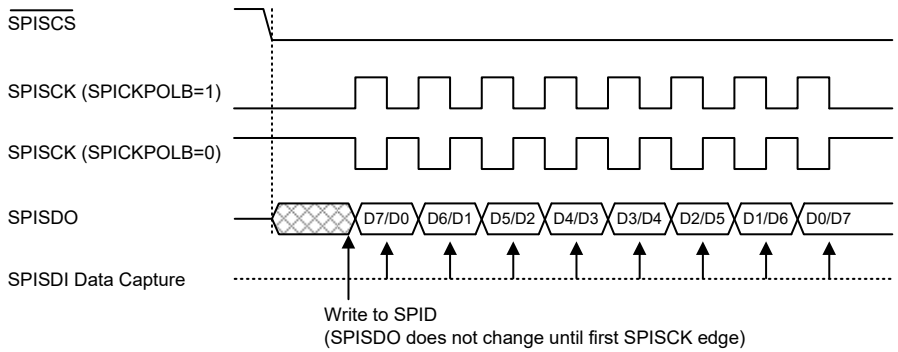
After the SPI interface is enabled by setting the SPIEN bit high, then in the Master Mode, when data is written to the SPID register, transmission/reception will begin simultaneously. When the data transfer is complete, the SPITRF flag will be set automatically, but must be cleared using the application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

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

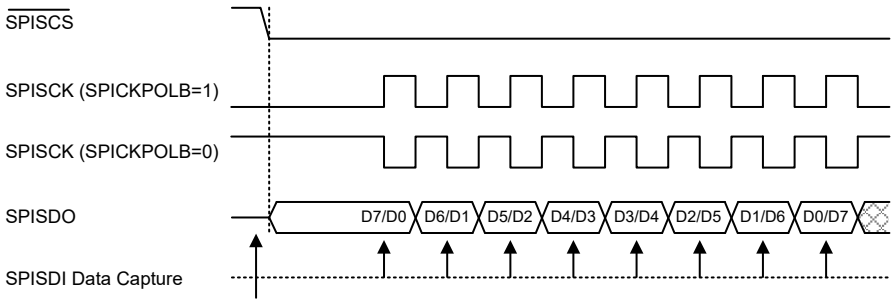
The SPI will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.



SPI Master Mode Timing

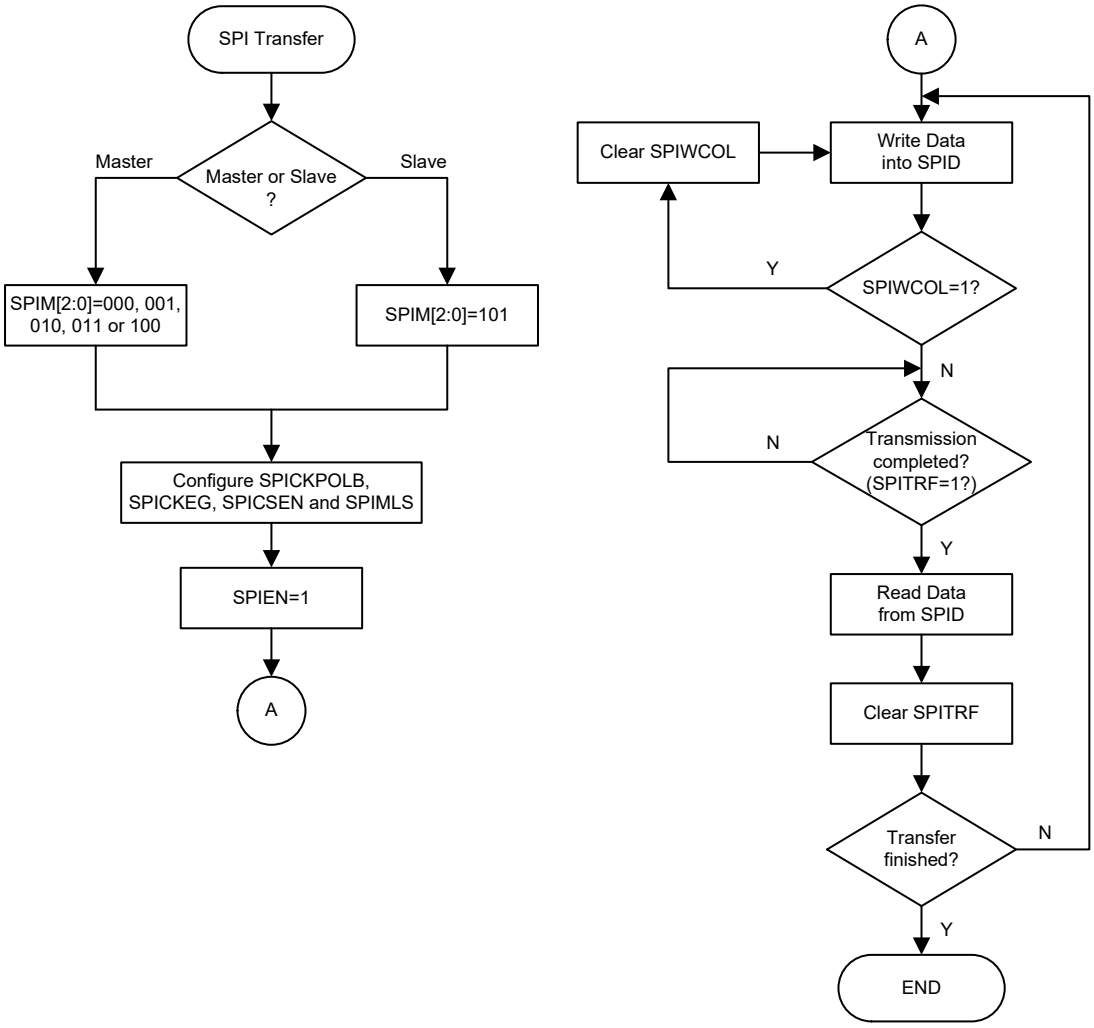


SPI Slave Mode Timing – SPICKEG=0



Note: For SPI slave mode, if SPIEN=1 and SPICSEN=0, SPI is always enabled and ignores the SPISCS level.

SPI Slave Mode Timing – SPICKEG=1



SPI Transfer Control Flowchart

SPI Bus Enable/Disable

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

When the SPI bus is diabled, SPISCK, SPISDI, SPISDO, $\overline{\text{SPISCS}}$ will become I/O pins or the other pin-shared functions by configuring the corresponding pin-shared selection bits.

SPI Operation Steps

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

The SPICSEN bit in the SPIC1 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{\text{SPISCS}}$ line to be active, which can then be used to control the SPI interface. If the SPICSEN bit is low, the SPI interface will be disabled and the $\overline{\text{SPISCS}}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the SPICSEN bit and the SPIEN bit in the SPIC0 register are set high, this will place the SPISDI line in a floating condition and the SPISDO line high. If in Master Mode the SPISCK line will be either high or low depending upon the clock polarity selection bit SPICKPOLB in the SPIC1 register. If in Slave Mode the SPISCK line will be in a floating condition. If SPIEN is low then the bus will be disabled and $\overline{\text{SPISCS}}$, SPISDI, SPISDO and SPISCK will all become I/O pins or the other functions using the corresponding pin-shared function selection 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 SPID 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 SPIM2~SPIM0 bits in the SPIC0 control register.
- Step 2
  Setup the SPICSEN bit and setup the SPIMLS bit to choose if the data is MSB or LSB first, this must be same as the Slave device.
- Step 3
  Setup the SPIEN bit in the SPIC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SPID register, which will actually place the data into the TXRX buffer. Then use the SPISCK and $\overline{\text{SPISCS}}$ lines to output the data. After this go to step 5.
  For read operations: the data transferred in on the SPISDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPID register.
- Step 5
  Check the SPIWCOL 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 SPITRF bit or wait for a SPI serial bus interrupt.
- Step 7
  Read data from the SPID register.

- Step 8
  Clear SPITRF.
- Step 9
  Go to step 4.

Slave Mode

- Step 1
  Select the SPI Slave mode using the SPIM2~SPIM0 bits in the SPIC0 control register.
- Step 2
  Setup the SPICSEN bit and setup the SPIMLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master device.
- Step 3
  Setup the SPIEN bit in the SPIC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SPID register, which will actually place the data into the TXRX buffer. Then wait for the master clock SPISCK and $\overline{\text{SPISCS}}$ signal. After this, go to step 5.
  For read operations: the data transferred in on the SPISDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPID register.
- Step 5
  Check the SPIWCOL 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 SPITRF bit or wait for a SPI serial bus interrupt.
- Step 7
  Read data from the SPID register.
- Step 8
  Clear SPITRF.
- Step 9
  Go to step 4.

Error Detection

The SPIWCOL bit in the SPIC1 register is provided to indicate errors during data transfer. The bit is set by the SPI serial Interface but must be cleared by the application program. This bit indicates a data collision has occurred which happens if a write to the SPID register takes place during a data transfer operation and will prevent the write operation from continuing.

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.

Driver No.	Duty	Bias Level	Bias Type	Waveform Type
36×4	1/4	1/3	C type	A or B
34×6	1/6	1/3	C type	A or B
32×8	1/8	1/3	C type	A or B

LCD Driver Output Selection

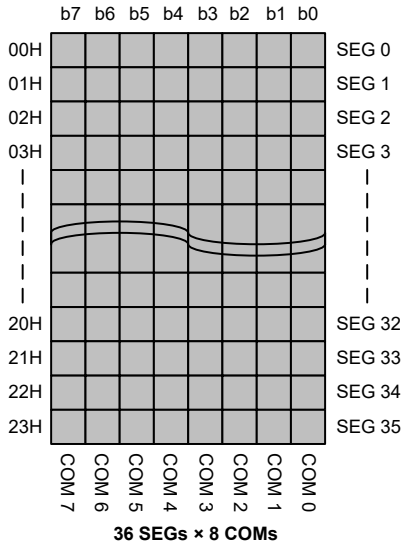
LCD Display Data Memory

An area of Data Memory is especially reserved for use for the LCD display data. This data area is known as the LCD Display Data 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.

The device provides an area of embedded data memory for the LCD display. The area is located at 00H to 23H in Sector 4 of the Data Memory. The LCD display memory can be read and written by indirect addressing mode using MP1L/MP1H and MP2L/MP2H, or by direct addressing mode using the corresponding extended instructions. Using the indirect addressing to access the LCD Display Data 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 23H, 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 accompanying LCD Memory Map diagrams shows how the internal LCD Memory is mapped to the Segments and Commons of the display 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 (4COM), the COM b4~b7 will be read as 0 only.



36 SEGs × 8 COMs

LCD Memory Map

LCD Clock Source

The LCD clock source is the internal clock signal, f_{SUB}, divided by 8 using an internal divider circuit. The f_{SUB} internal clock is supplied by the LIRC or LXT oscillator, 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 source frequency of 4 kHz.

LCD Register Description

Control registers in the Data Memory are used to control the vatious setup features of the LCD Driver. There are two control registers for the LCD function, LCDC0 and LCDC2.

The TYPE bit in the LCDC0 register is used to select whether Type A or Type B LCD control signals are used. Bits LCDP1 and LCDP0 in the LCDC0 register are used to select the power source to supply the C type LCD panel with the correct bias voltages. The LCDEN bit in the LCDC0 register, which provides the overall LCD enable/disable function, will only be effective when this device is in the FAST, SLOW or IDLE Mode. If this device is in the SLEEP Mode then the display will always be disabled.

The LCDC2 register is used for the C-type LCD pump clock and LCD duty selection.

Register Name	Bit							
	7	6	5	4	3	2	1	0
LCDC0	TYPE	—	LCDP1	LCDP0	—	—	—	LCDEN
LCDC2	LCDPCK2	LCDPCK1	LCDPCK0	—	—	DTYC1	DTYC0	—

LCD Driver Register List

- **LCDC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	TYPE	—	LCDP1	LCDP0	—	—	—	LCDEN
R/W	R/W	—	R/W	R/W	—	—	—	R/W
POR	0	—	0	0	—	—	—	0

Bit 7 **TYPE**: LCD waveform type selection
0: Type A
1: Type B

Bit 6 Unimplemented, read as "0"

Bit 5~4 **LCDP1~LCDP0**: C type LCD power source selection
00: From external pin PLCD/V1/V2
01: From internal reference voltage V_{REFIN} supplied to V_C
10: From internal voltage 3V supplied to V_B
11: From internal voltage V_{DD} supplied to V_A
The V_{REFIN} is an internal reference voltage with an approximate level of 1.0V.

Bit 3~1 Unimplemented, read as "0"

Bit 0 **LCDEN**: LCD Enable Control
0: Disable
1: Enable
In the FAST, SLOW or IDLE mode, the LCD on/off function can be controlled by this bit. However, in the SLEEP mode, the LCD function is always switched off.

• **LCDC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	LCDPCK2	LCDPCK1	LCDPCK0	—	—	DTYC1	DTYC0	—
R/W	R/W	R/W	R/W	—	—	R/W	R/W	—
POR	0	0	0	—	—	0	0	—

Bit 7~5 **LCDPCK2~LCDPCK0**: C-type LCD Pump clock divider selection
000: 250Hz (f_{SUB}/128)
001: 500Hz (f_{SUB}/64)
010: 1kHz (f_{SUB}/32)
011: 2kHz (f_{SUB}/16)
100: 4kHz (f_{SUB}/8)
101: 8kHz (f_{SUB}/4)
110: 16kHz (f_{SUB}/2)
111: 16kHz (f_{SUB}/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 used)
01: 1/6 Duty (COM0~COM5 used)
10: 1/8 Duty (COM0~COM7 used)
11: Unimplemented
The unused COM pins are allowed to be configured as normal I/O or other pin-shared functions.

Bit 0 Unimplemented, read as "0"

LCD Voltage Source and Biasing

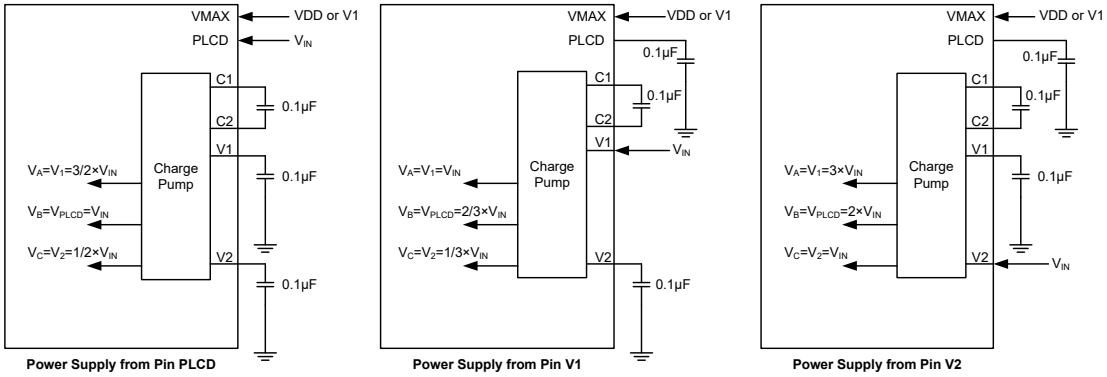
The LCD voltage source can be supplied on the external pin PLCD, V1 or V2 or derived from the internal voltage source to generate the required biasing voltages. The C type bias voltage source is selected using the LCDP1 and LCDP0 bits in the LCDC0 register. When the LCD voltage source is from the PLCD or V2 pin, the C type biasing scheme uses an internal charge pump circuit and can generate voltages higher than what is supplied on the pin. This feature is useful in applications where the microcontroller supply voltage is less than the supply voltage required by the LCD. The charge pump clock source is the internal clock signal, f_{SUB}, which is divided by an internal divider circuit. The divider value is selected using the LCDPCK2~LCDPCK0 bits in the LCDC2 register. 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 and 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.

LCD Power Supply		V_A voltage	V_B voltage	V_C voltage
External Power Supply	V_{IN} From V1 pin	V_{IN}	2/3 × V_{IN}	1/3×V_{IN}
	V_{IN} From PLCD pin	3/2×V_{IN}	V_{IN}	1/2×V_{IN}
	V_{IN} From V2 pin	3×V_{IN}	2×V_{IN}	V_{IN}
Internal Power Supply	V_A (V_A=V_{DD})	V_{DD}	2/3×V_{DD}	1/3×V_{DD}
	V_B (V_B=3V)	3/2×3V	3V	1/2×3V
	V_C (V_C=V_{REFIN})Note	3×V_{REFIN}	2×V_{REFIN}	V_{REFIN}

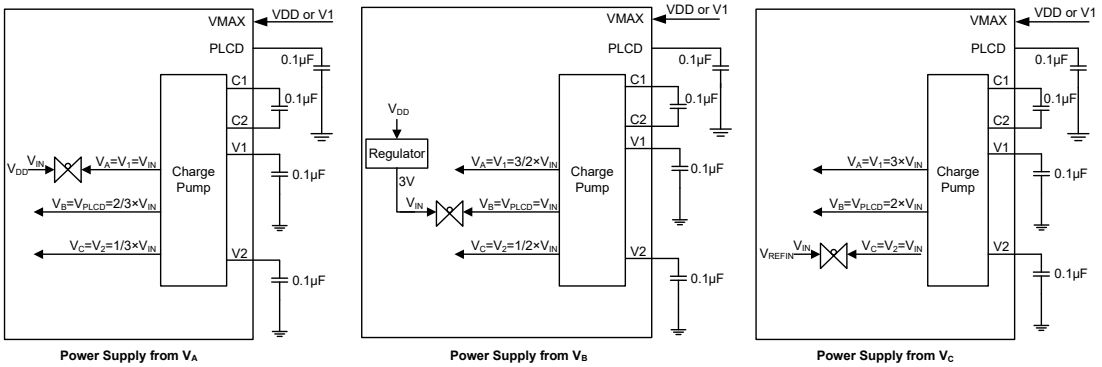
Note: The V_{REFIN} voltage is from the device ingerated depletion circuit and has an approximate level of 1.0V.

C Type Bias Power Supply Scheme



Note: The pin VMAX must be connected to the maximum voltage to prevent from the pad current leakage.

C Type Bias External Power Supply Configuration – 1/3 Bias



Note: The pin VMAX must be connected to the maximum voltage to prevent from the pad current leakage.

C Type Bias Internal Power Supply Configuration – 1/3 Bias

The connection to the VMAX pin depends upon the LCD power supply scheme. It is extremely important to ensure that these charge pump generated internal voltages do not exceed the maximum V_{DD} voltage of 5.5V.

Condition	VMAX Connection
$V_{DD} > V_{PLCD} \times 1.5$	Connect VMAX to VDD
Otherwise	Connect VMAX to V1

C Type Bias VMAX Pin Connection

LCD Reset Status

The LCD has an internal reset function that is an OR function of the inverted LCDEN bit in the LCDC0 register and the SLEEP function. Clearing the LCDEN bit to zero will reset the LCD function. The LCD function will also be reset after the device enters the SLEEP mode even if the LCDEN bit is set to "1" to enable the LCD driver 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 output 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	SLEEP Mode	LCDEN	LCD Reset	COM & SEG Voltage Level
No	Off	1	No	Normal Operation
No	Off	0	Yes	Low
No	On	x	Yes	Low
Yes	x	x	Yes	Floating

Note: 1. The watchdog time-out reset in the IDLE or SLEEP Mode is excluded from the MCU Reset conditions.
2. "x": Don't care.

LCD Reset Status

LCD Driver Output

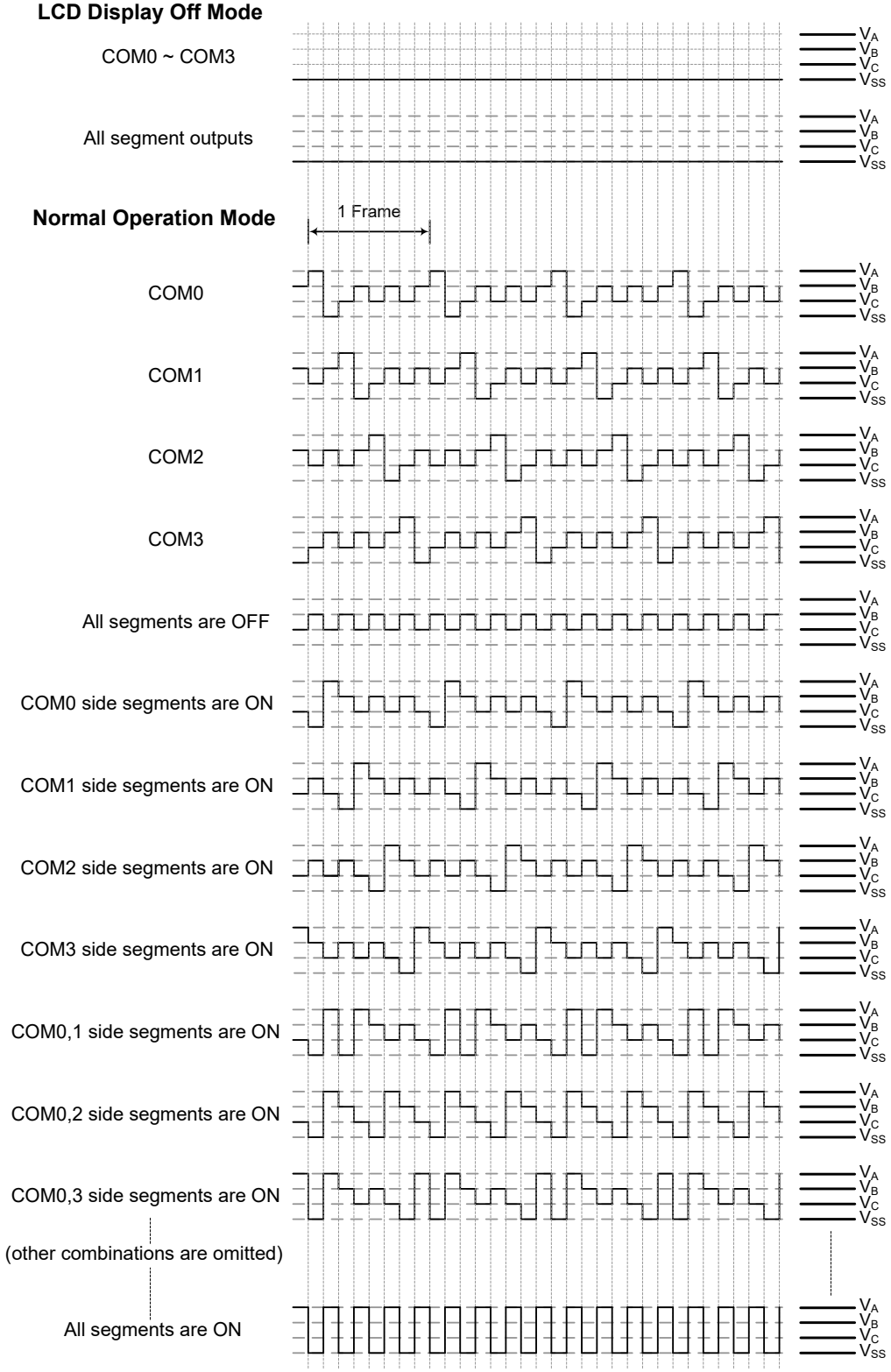
The number of COM and SEG outputs supplied by the LCD driver, as well as its biasing and waveform type selections, are dependent upon how the LCD control bits are programmed. The LCD driver bias type is C type and has a fixed bias of 1/3.

The nature of Liquid Crystal Displays requires 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.

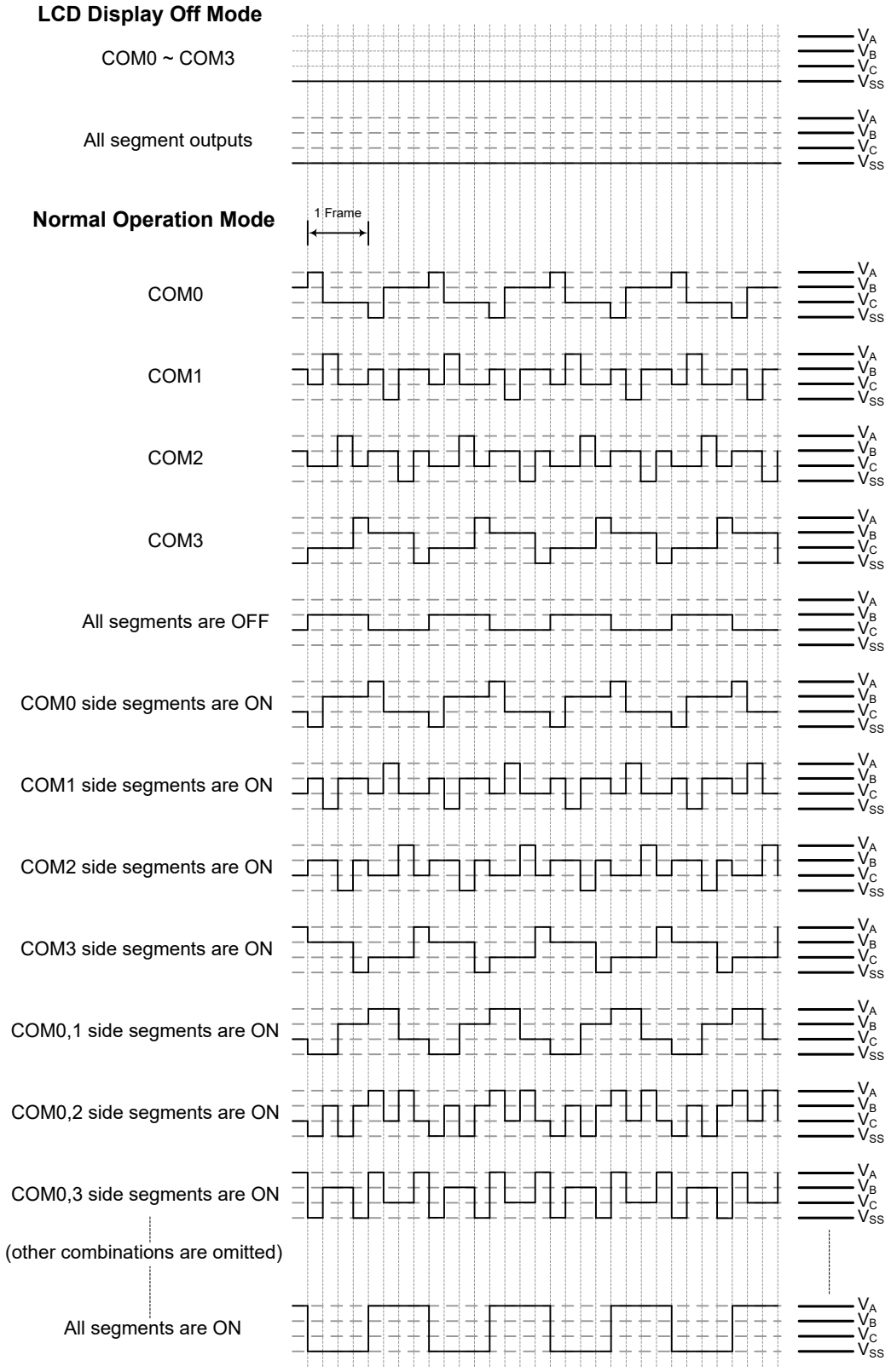
The requirement to limit the DC voltage to zero and to control as many pixels as possible with a minimum number of connections requires that both a time and amplitude signal is generated and applied to the application LCD. These time and amplitude varying signals are automatically generated by the LCD 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, which is to have a value of 1/4 and which equates to a COM number of 4, therefore defines the number of time divisions within each LCD signal frame. Two types of signal generation are also provided, known as Type A and Type B, the required type is selected via the TYPE bit in the LCDC0 register. Type B offers lower frequency signals, however, lower frequencies may introduce flickering and influence display clarity.

C Type, 4 COM, 1/3 Bias

LCD Display Off Mode

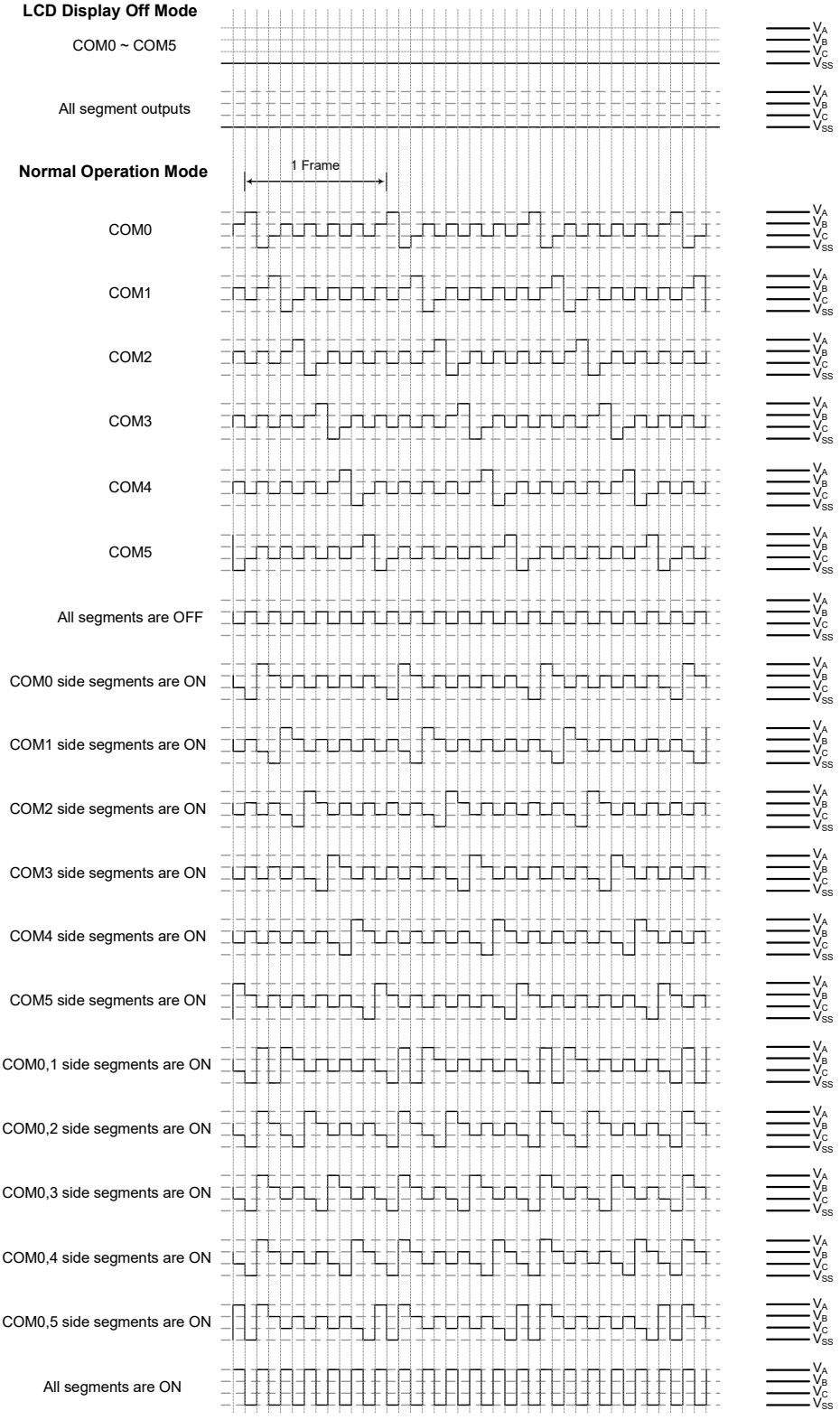


LCD Driver Output – Type A, 1/4 duty, 1/3 bias

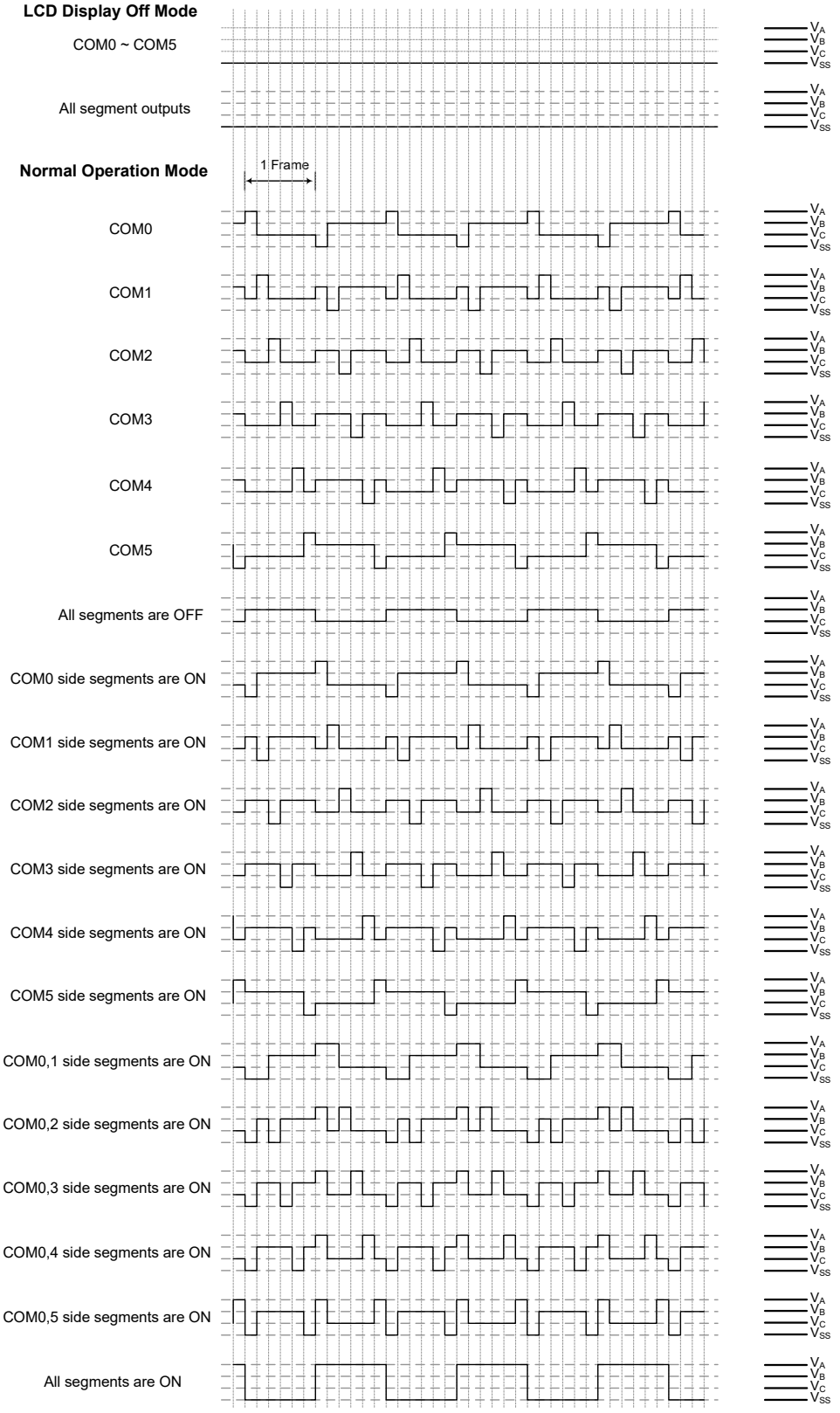


LCD Driver Output – Type B, 1/4 duty, 1/3 bias

C Type, 6 COM, 1/3 Bias

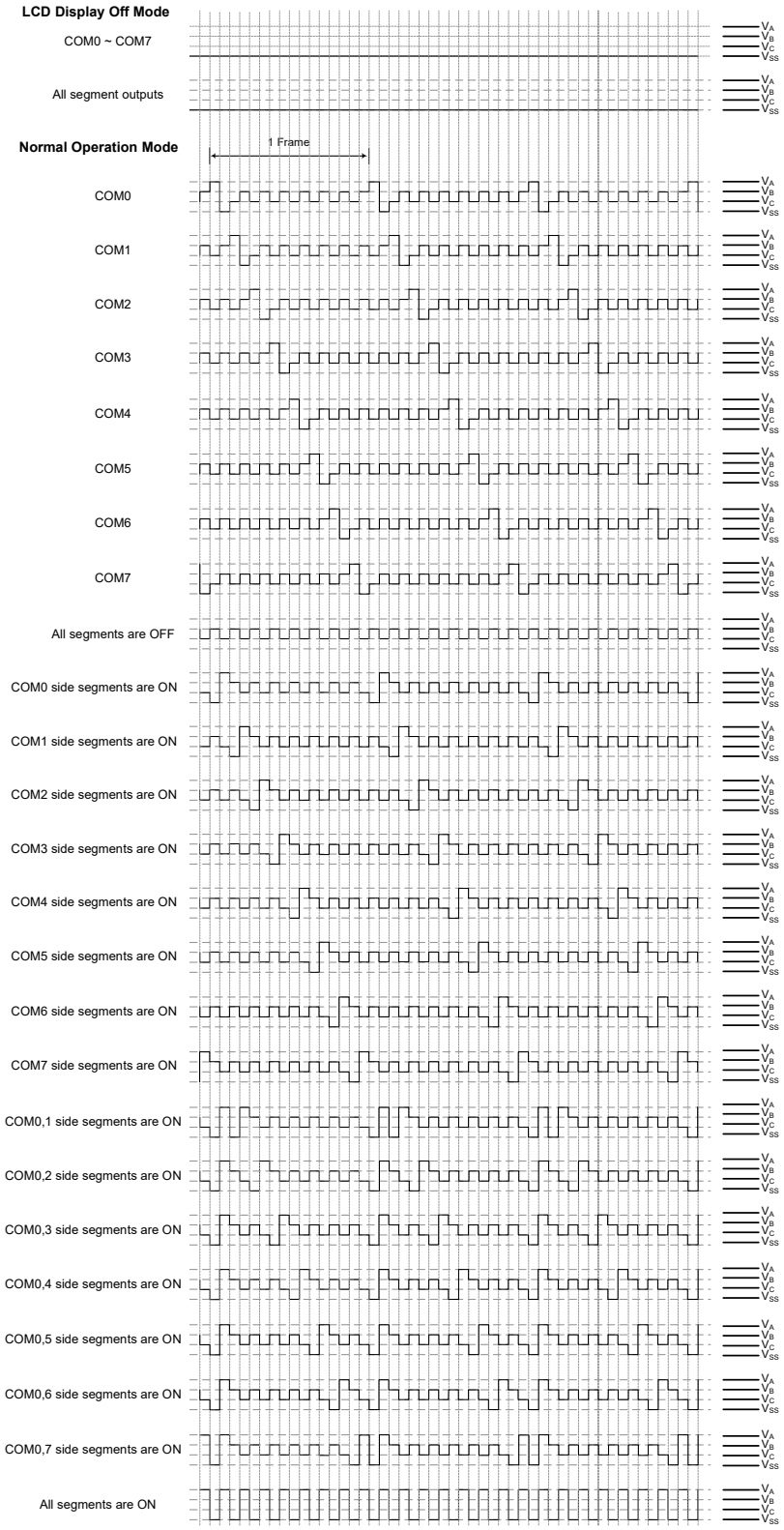


LCD Driver Output – Type A, 1/6 duty, 1/3 bias

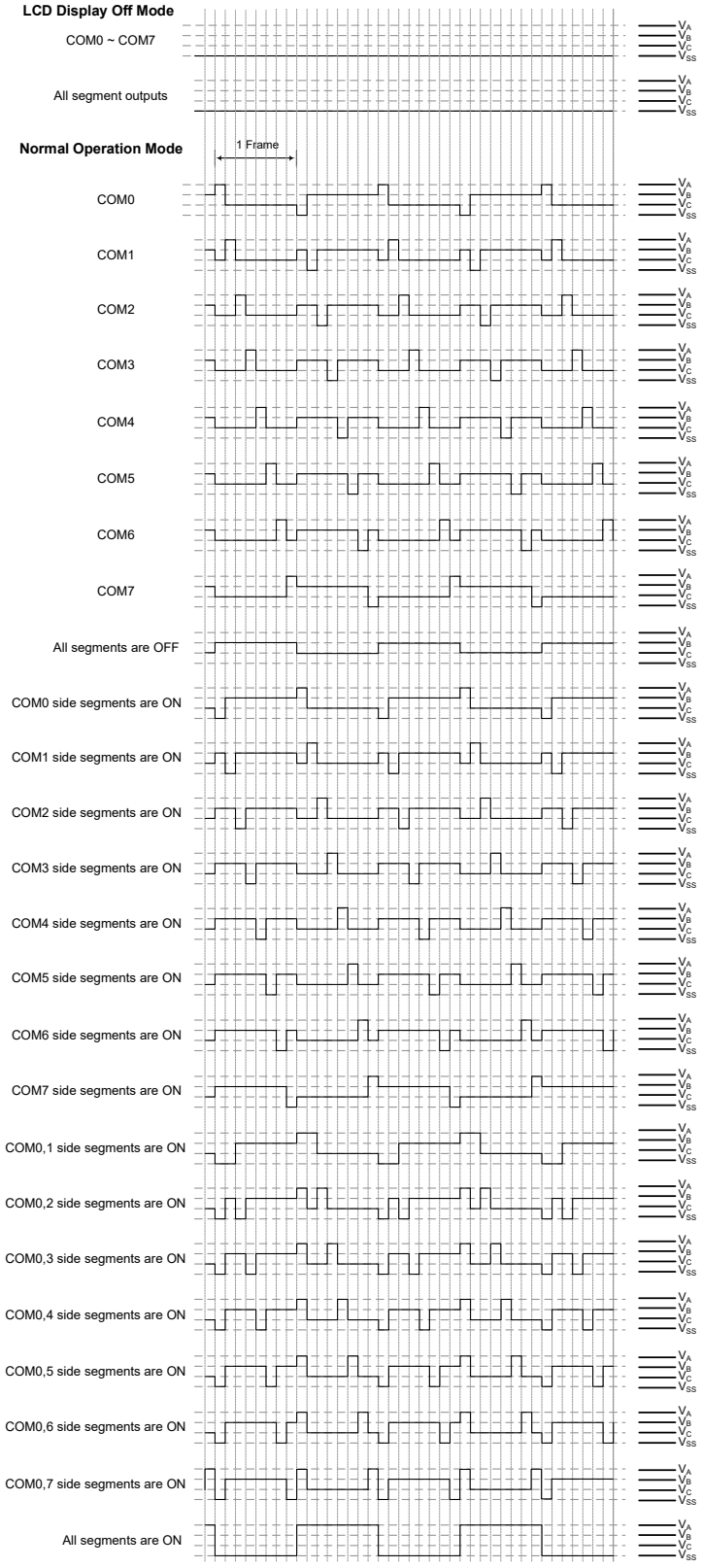


LCD Driver Output – Type B, 1/6 duty, 1/3 bias

C 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

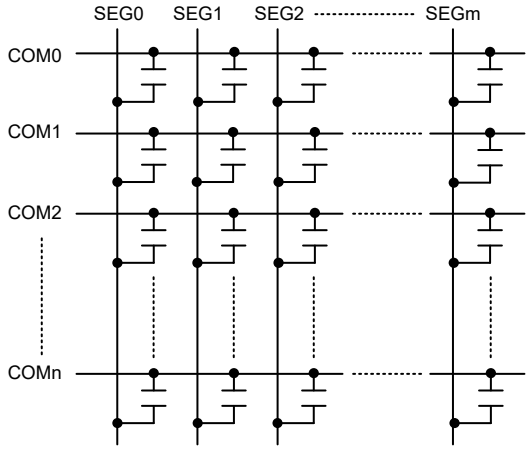
Programming Considerations

Certain precautions must be taken when programming the LCD. One of these is to ensure that the LCD Memory is properly initialised after the microcontroller is powered on. Like the General Purpose Data Memory, the contents of the LCD Memory are in an unknown condition after power-on. As the contents of the LCD Memory will be mapped into the actual display, it is important to initialise this memory area into a known condition soon after applying power to obtain a proper display pattern.

Consideration must also be given to the capacitive load of the actual LCD used in the application. As the load presented to the microcontroller by LCD pixels can be generally modeled as mainly capacitive in nature, it is important that this is not excessive, a point that is particularly true in the case of the COM lines which may be connected to many LCD pixels. The accompanying diagram depicts the equivalent circuit of the LCD.

One additional consideration that must be taken into account is what happens when the microcontroller enters the IDLE or SLOW Mode. The LCDEN control bit in the LCDC0 register permits the display to be powered off to reduce power consumption. If this bit is zero, the driving signals to the display will cease, producing a blank display pattern but reducing any power consumption associated with the LCD.

After Power-on, note that as the LCDEN bit will be cleared to zero, the display function will be disabled.



LCD Panel Equivalent Circuit

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions. The external interrupts are generated by the action of the external INT0~INT3 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, EEPROM, USIMs, SPI and the A/D converter, etc.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers depends upon the device chosen but fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI2 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

Each register contains a number of enable bits to enable or disable individual interrupts 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 Pins	INTnE	INTnF	n=0~3
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~2
Time Base	TBnE	TBnF	n=0~1
USIM Interface	USIMnE	USIMnF	n=0~1
SPI	SPIE	SPIF	—
EEPROM erase/write operation	DEE	DEF	—
PTM	PTMnPE	PTMnPF	n=0~1
	PTMnAE	PTMnAF	
STM	STMPE	STMPF	—
	STMAE	STMAF	
ATM	ATMPE	ATMPF	—
	ATMAE	ATMAF	
	ATMBE	ATMBF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
INTC1	INT2F	MF2F	MF1F	MF0F	INT2E	MF2E	MF1E	MF0E
INTC2	USIM0F	TB1F	TB0F	INT3F	USIM0E	TB1E	TB0E	INT3E
INTC3	—	—	SPIF	USIM1F	—	—	SPIE	USIM1E
MFI0	PTM1AF	PTM1PF	PTM0AF	PTM0PF	PTM1AE	PTM1PE	PTM0AE	PTM0PE
MFI1	—	DEF	STMAF	STMPF	—	DEE	STMAE	STMPE
MFI2	—	ATMBF	ATMAF	ATMPF	—	ATMBE	ATMAE	ATMPE

Interrupt Register

• **ListINTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **INT3S1~INT3S0**: Interrupt edge control for INT3 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 5~4 **INT2S1~INT2S0**: Interrupt edge control for INT2 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 3~2 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt edge control for INT0 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 5 **INT1F**: 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 **ADE**: A/D Converter 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	INT2F	MF2F	MF1F	MF0F	INT2E	MF2E	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 **INT2F**: INT2 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **MF2F**: Multi-function 2 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **MF1F**: Multi-function 1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **MF0F**: Multi-function 0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **INT2E**: INT2 interrupt control
0: Disable
1: Enable

Bit 2 **MF2E**: Multi-function 2 interrupt control
0: Disable
1: Enable

Bit 1 **MF1E**: Multi-function 1 interrupt control
0: Disable
1: Enable

Bit 0 **MF0E**: Multi-function 0 interrupt control
0: Disable
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	USIM0F	TB1F	TB0F	INT3F	USIM0E	TB1E	TB0E	INT3E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **USIM0F**: USIM0 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INT3F**: INT3 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **USIM0E**: USIM0 interrupt control
0: Disable
1: Enable

Bit 2 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

Bit 1 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

Bit 0 **INT3E**: INT3 interrupt control
0: Disable
1: Enable

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SPIF	USIM1F	—	—	SPIE	USIM1E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **SPIF**: SPI interrupt request flag
0: No request
1: Interrupt request

Bit 4 **USIM1F**: USIM1 interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **SPIE**: SPI interrupt control
0: Disable
1: Enable

Bit 0 **USIM1E**: USIM1 interrupt control
0: Disable
1: Enable

• **MFI0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTM1AF	PTM1PF	PTM0AF	PTM0PF	PTM1AE	PTM1PE	PTM0AE	PTM0PE
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 **PTM1AF**: PTM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **PTM1PF**: PTM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 5 **PTM0AF**: PTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTM0PF**: PTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 **PTM1AE**: PTM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **PTM1PE**: PTM1 Comparator P match 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

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	DEF	STMAF	STMPF	—	DEE	STMAE	STMPE
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 **DEF**: Data EEPROM interrupt request flag
0: No request
1: Interrupt request

Bit 5 **STMAF**: STM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **STMPF**: STM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **DEE**: Data EEPROM interrupt control
0: Disable
1: Enable

Bit 1 **STMAE**: STM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **STMPE**: STM Comparator P match interrupt control
0: Disable
1: Enable

• **MFI2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	ATMBF	ATMAF	ATMPF	—	ATMBE	ATMAE	ATMPE
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 **ATMBF**: ATM Comparator B match interrupt request flag
0: No request
1: Interrupt request

Bit 5 **ATMAF**: ATM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **ATMPF**: ATM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3	Unimplemented, read as "0"
Bit 2	**ATMBE**: ATM Comparator B match interrupt control 0: Disable 1: Enable
Bit 1	**ATMAE**: ATM Comparator A match interrupt control 0: Disable 1: Enable
Bit 0	**ATMPE**: ATM Comparator P match interrupt control 0: Disable 1: Enable

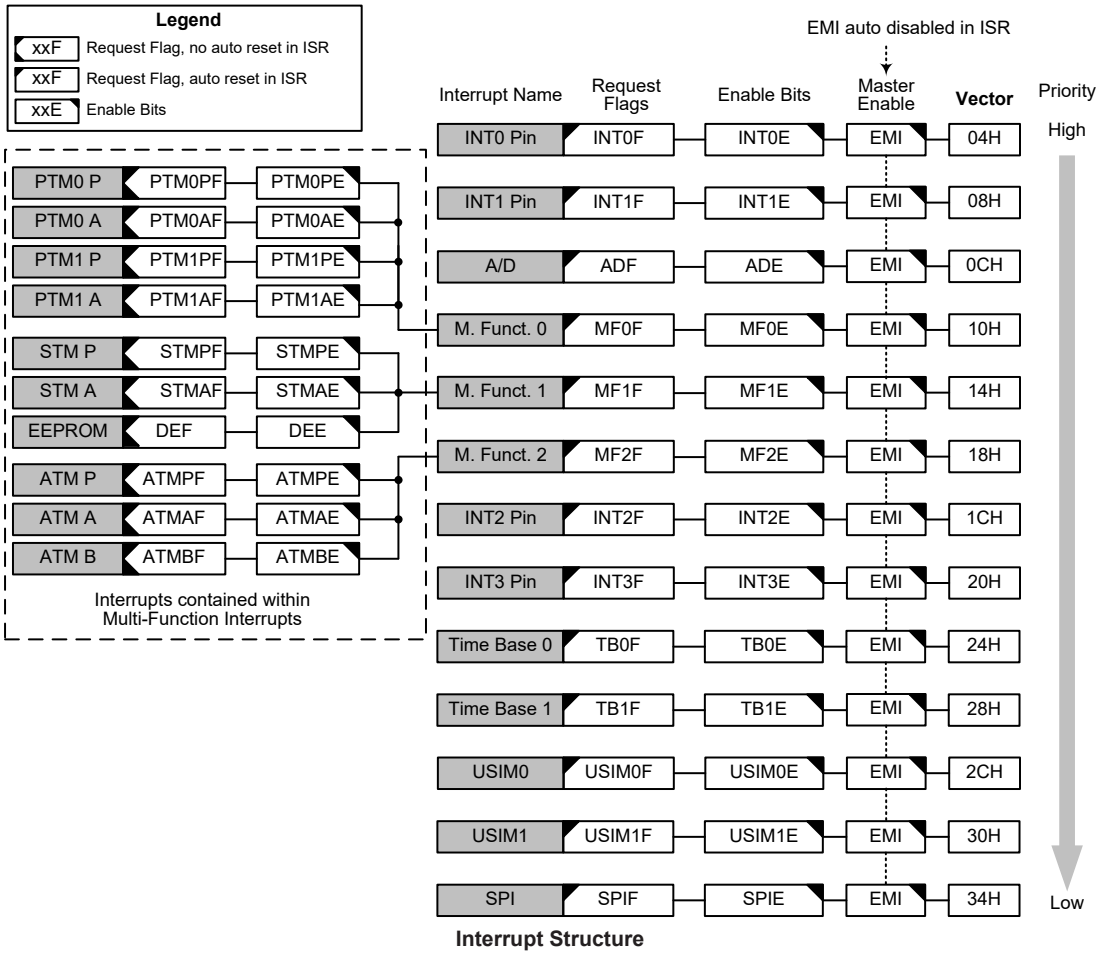
Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A 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.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.



Interrupt Structure

External Interrupts

The external interrupts are controlled by signal transitions on the pins INT0~INT3. An external interrupt request will take place when the external interrupt request flags, INT0F~INT3F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT3E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT3F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

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

A/D Converter Interrupt

The A/D Converter Interrupt is controlled by the termination of an A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter Interrupt vector, will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Multi-function Interrupts

Within the device there are three Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts and EEPROM erase or write complete interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags MFnF is set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. When the Multi-function interrupt is enabled and the stack is not full, and one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function Interrupt 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.

Timer Module Interrupts

The Standard and Periodic type TMs each has two interrupts, while the Audio Type TM has three interrupts. All of these TM interrupts are contained within the Multi-function Interrupts. For each of the standard and Periodic Type TMs there are two interrupt request flags and two interrupt enable bits. For the Audio Type TM there are three interrupt request flages and three 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, A or B match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

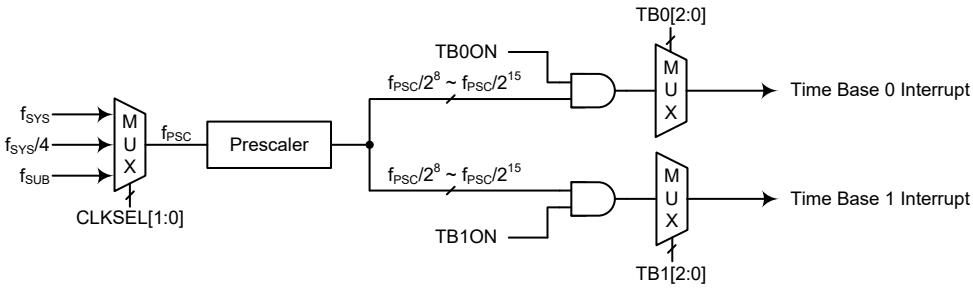
EEPROM Interrupt

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 its respective interrupt vector address, the global interrupt enable bit, EMI, EEPROM Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and an EEPROM erase or write cycle ends, a subroutine call to the relevant 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 will be automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signals in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TBnF, will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, and Time Base enable bits, TBnE, 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, TBnF, will be automatically cleared, the EMI bit will also be automatically cleared to disable other interrupts.

The purpose of the Time Base Interrupts is to provide an interrupt signal at fixed time periods. Their clock source, f_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBnC register to obtain longer interrupt periods whose value ranges. The clock cource which in turn controls the Time Base interrupt period is selected using the CLKSEL[1:0] bits in the PSCR register.



Time Base Interrupts

- **PSCR Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source f_{PSC} selection
00: f_{SYS}
01: $f_{SYS}/4$
10/11: f_{SUB}

• **TB0C Register**

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

Bit 7 **TB0ON**: Time Base 0 Enable Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Time Base 0 time-out period selection
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

• **TB1C Register**

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

Bit 7 **TB1ON**: Time Base 1 Enable Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB12~TB10**: Time Base 1 time-out period selection
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

Universal Serial Interface Module Interrupts

The Universal Serial Interface Module Interrupts, also known as the USIM0 and USIM1 interrupts, will take place when the USIMn Interrupt request flag, USIMnF, is set. As the USIMn interface can operate in three modes which are SPI mode, I^2C mode and UART mode, the USIMnF flag can be set by different conditions depending on the selected interface mode.

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

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, USIMnE, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the respective Interrupt vector, will take place. When the interrupt is serviced, the Universal Serial Interface Interrupt flag, USIMnF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Note that if the USIMn interrupt is triggered by the UART interface, after the interrupt has been serviced, the UnUSR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART Interface content of the USIM chapter.

SPI Interface Interrupt

An SPI Interrupt request will take place when the SPI Interrupt request flag, SPIF, is set, which occurs when a byte of data has been received or transmitted by the SPI interface. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, SPIE, must first be set. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SPI interface, a subroutine call to the respective Interrupt vector, will take place. When the SPI Interface Interrupt is serviced, the SPI interrupt request flag, SPIF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

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.

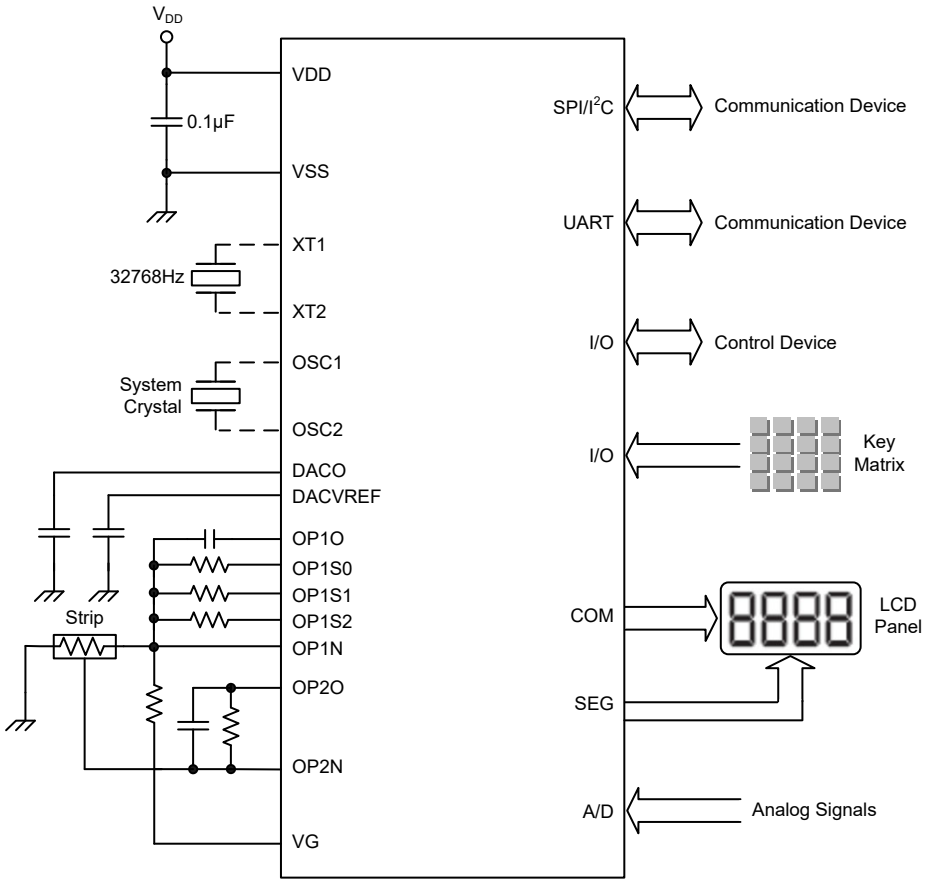
Configuration Options

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}: 4MHz, 8MHz or 12MHz

Note: When the HIRC has been configured at a frequency shown in this table, the HIRC1 and HIRC0 bits should also be setup to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Application Circuits



Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs 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] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

DEC [m] Decrement Data Memory

Description: Data in the specified Data Memory is decremented by 1.

Operation: [m] ← [m] − 1

Affected flag(s): Z

DECA [m] Decrement Data Memory with result in ACC

Description: Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation: ACC ← [m] − 1

Affected flag(s): Z

HALT Enter power down mode

Description: This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation: TO ← 0
PDF ← 1

Affected flag(s): TO, PDF

INC [m] Increment Data Memory

Description: Data in the specified Data Memory is incremented by 1.

Operation: [m] ← [m] + 1

Affected flag(s): Z

INCA [m] Increment Data Memory with result in ACC

Description: Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation: ACC ← [m] + 1

Affected flag(s): Z

JMP addr Jump unconditionally

Description: The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation: Program Counter ← addr

Affected flag(s): None

MOV A,[m] Move Data Memory to ACC

Description: The contents of the specified Data Memory are copied to the Accumulator.

Operation: ACC ← [m]

Affected flag(s): None

MOV A,x Move immediate data to ACC

Description: The immediate data specified is loaded into the Accumulator.

Operation: ACC ← x

Affected flag(s): None

MOV [m],A Move ACC to Data Memory

Description: The contents of the Accumulator are copied to the specified Data Memory.

Operation: [m] ← ACC

Affected flag(s): None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

OR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" x

Affected flag(s) Z

ORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

RET Return from subroutine

Description The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation Program Counter ← Stack

Affected flag(s) None

RET A,x Return from subroutine and load immediate data to ACC

Description The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

Description The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

RLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

RR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory is rotated right by 1 bit with bit 0
rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s): C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: $ACC \leftarrow ACC - [m] - \overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

SBC A, x Subtract immediate data from ACC with Carry

Description: The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: $ACC \leftarrow ACC - [m] - \overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: $[m] \leftarrow ACC - [m] - \overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

SDZ [m] Skip if decrement Data Memory is 0

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation:
[m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation:
ACC ← [m] − 1
Skip if ACC=0

Affected flag(s): None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

SNZ [m].i Skip if 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 $[m] \leftarrow \overline{[m]}$

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow \overline{[m]}$

Affected flag(s) Z

LDAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

LDEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation $[m] \leftarrow [m] - 1$

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

LINC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation $[m] \leftarrow [m] + 1$

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] + 1$

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

LMOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

LOR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

LORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

LRL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

LRLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

LRLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

LRLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

LRR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

LRRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s) None

LRRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

LRRCA [m] Rotate Data Memory right through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s) C

LSBC A,[m] Subtract Data Memory from ACC with Carry

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m] − $\overline{C}$

Affected flag(s) OV, Z, AC, C, SC, CZ

LSBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m] − $\overline{C}$

Affected flag(s) OV, Z, AC, C, SC, CZ

LSDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a 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

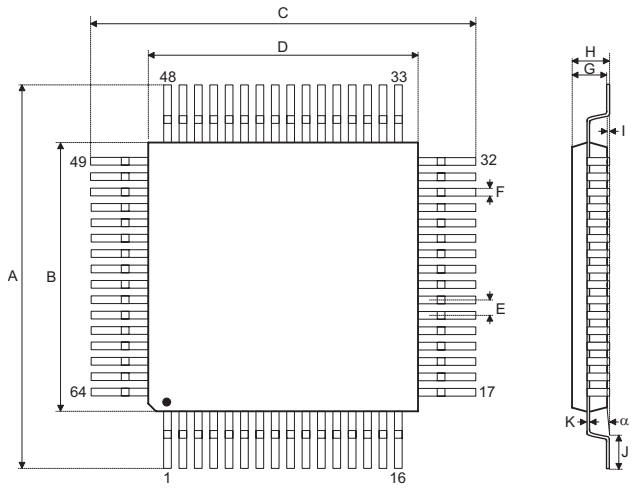
Package Information

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

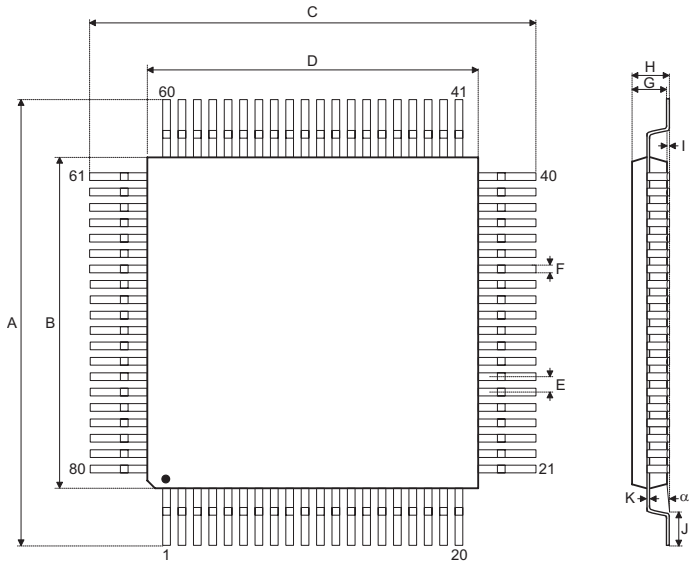
64-pin LQFP (7mm×7mm) Outline Dimensions



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°

80-pin LQFP (10mm×10mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.472 BSC	
B		0.394 BSC	
C		0.472 BSC	
D		0.394 BSC	
E		0.016 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		12.00 BSC	
B		10.00 BSC	
C		12.00 BSC	
D		10.00 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°