



Touch 24-Bit A/D Flash MCU

BS45F6052

Revision: V1.30 Date: December 03, 2025

www.holtek.com

Features

CPU Features

- Operating voltage
 - ♦ $f_{SYS}=8\text{MHz}$: 1.8V~5.5V
- Up to 0.5 μs instruction cycle with 8MHz system clock
- Power down and wake-up functions to reduce power consumption
- Oscillator types
 - ♦ Internal High Speed 8MHz RC – HIRC
 - ♦ Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 12-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 8K \times 16
- Data Memory: 512 \times 8
- True EEPROM Memory: 512 \times 8
- In Application Programming function – IAP
- 4 touch key functions – fully integrated without requiring external components
- 11 bidirectional I/O lines
- Two external interrupt lines shared with I/O pins
- One 8-bit programmable Timer with overflow interrupt and prescaler
- Two Timer Modules for time measure, compare match output, PWM output function
 - ♦ Two 10-bit Compact Type TMs – CTM0~CTM1
- Dual Time Base functions for generation of fixed time interrupt signals
- Fully-duplex/Half-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- I²C Interface
- Communication interfaces support VDDIO function
- 3 differential or 6 single-end external channel 24-bit resolution Delta Sigma A/D converter
- Internal high accuracy temperature sensor
- Watchdog Timer function
- Integrated 16-bit Cyclic Redundancy Check function – CRC
- Low voltage reset function
- Low voltage detect function
- Package type: 16-pin QFN

General Description

The device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller with all touch key functions provided internally, completely eliminating the need for external components. The device has all the features to offer designers a reliable and easy means of implementing touch keys within their products applications.

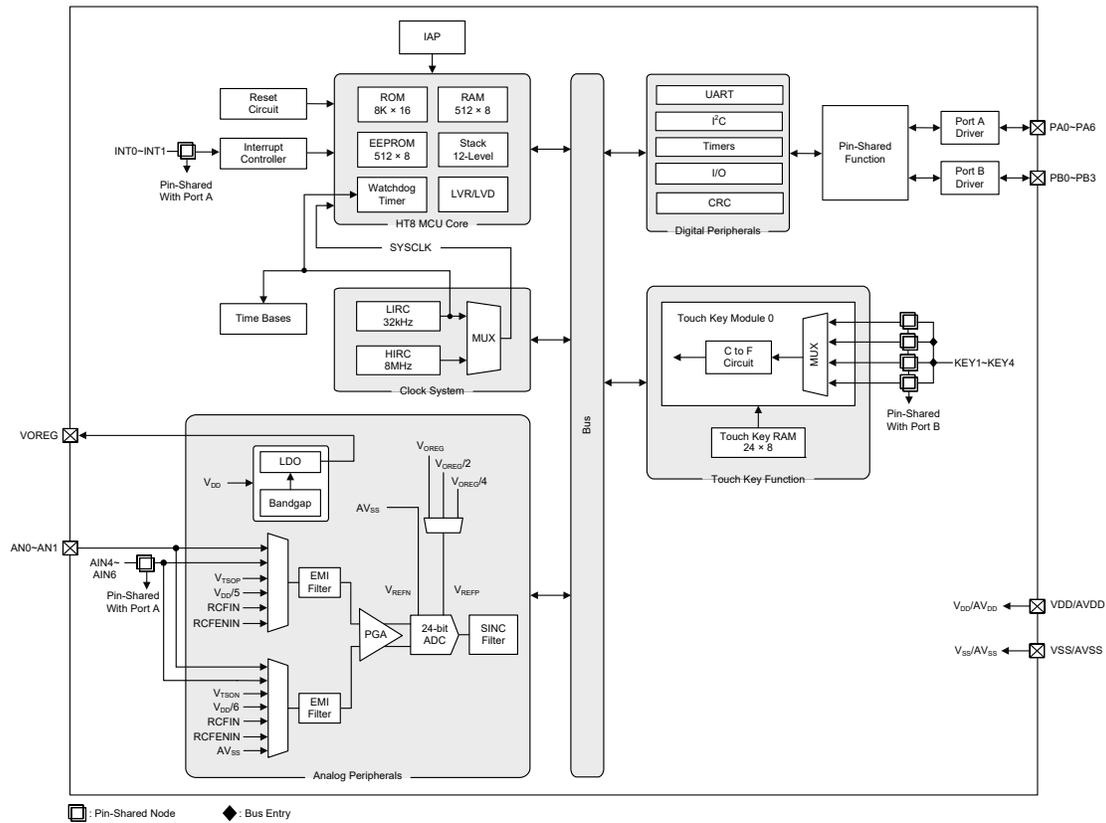
For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc. By using the In Application Programming 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 24-bit Delta Sigma A/D Converter with both single and differential inputs. Multiple and extremely flexible Timer Modules provide timing, compare match output and PWM generation functions. Communication with the outside world is catered for by including fully integrated I²C and UART interface functions, two popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

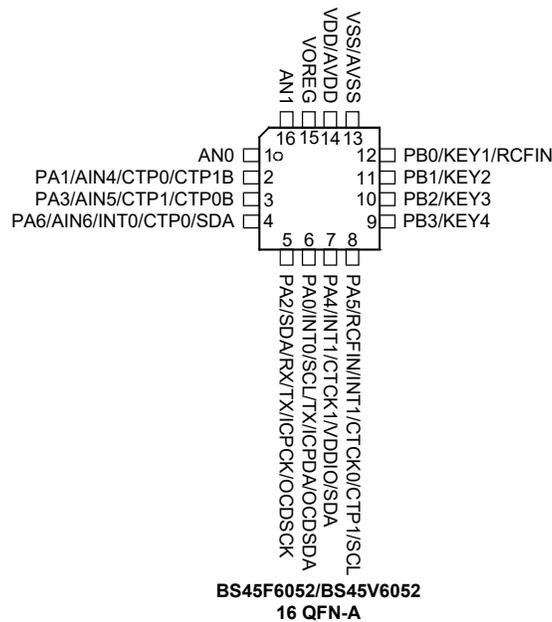
The device also includes fully integrated high and low speed oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

The inclusion of flexible I/O programming features, Time Base functions along with many other features enhance the versatility of the device to suit for applications that interface directly to analog signals which require a low noise and high accuracy analog to digital converter.

Block Diagram



Pin Assignment



- Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.
2. The OCSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the BS45V6052 device which is the OCDS EV chip for the BS45F6052 device.

Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT0/SCL/TX/ ICPDA/OCSDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT0	PAS0 INTEG INTC0 IFS	ST	—	External interrupt 0 input
	SCL	PAS0 IFS	ST	NMOS	I ² C clock line
	TX	PAS0	—	CMOS	UART serial data output
	ICPDA	—	ST	CMOS	ICP data/address pin
	OCSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
PA1/AIN4/CTP0/CTP1B	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	AIN4	PAS0	AN	—	ADC input channel 4
	CTP0	PAS0	—	CMOS	CTM0 output
	CTP1B	PAS0	—	CMOS	CTM1 inverted output

Pin Name	Function	OPT	I/T	O/T	Description
PA2/SDA/RX/TX/ICPCK/OCDSCK	PA2	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	SDA	PAS0 IFS	ST	NMOS	I ² C data line
	RX/TX	PAS0	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input / output in Single Wire Mode communication
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/AIN5/CTP1/CTP0B	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	AIN5	PAS0	AN	—	ADC input channel 5
	CTP1	PAS0	—	CMOS	CTM1 output
	CTP0B	PAS0	—	CMOS	CTM0 inverted output
PA4/INT1/CTCK1/VDDIO/SDA	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT1	PAS1 INTEG INTC3 IFS	ST	—	External interrupt 1 input
	CTCK1	PAS1	ST	—	CTM1 clock input
	VDDIO	PAS1	PWR	—	PA0, PA2 and PA5 pins power supply
	SDA	PAS1 IFS	ST	NMOS	I ² C data line
PA5/RCFIN/INT1/CTCK0/CTP1/SCL	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	RCFIN	PAS1	AN	—	ADC input channel with RC filter
	INT1	PAS1 INTEG INTC3 IFS	ST	—	External interrupt 1 input
	CTCK0	PAS1	ST	—	CTM0 clock input
	CTP1	PAS1	—	CMOS	CTM1 output
	SCL	PAS1 IFS	ST	NMOS	I ² C clock line
PA6/AIN6/INT0/CTP0/SDA	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	AIN6	PAS1	AN	—	ADC input channel 6
	INT0	PAS1 INTEG INTC0 IFS	ST	—	External interrupt 0 input
	CTP0	PAS0	—	CMOS	CTM0 output
	SDA	PAS1 IFS	ST	NMOS	I ² C data line
PB0/KEY1/RCFIN	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	KEY1	PBS0	AN	—	Touch key input 1
	RCFIN	PBS0	AN	—	ADC input channel with RC filter

Pin Name	Function	OPT	I/T	O/T	Description
PB1/KEY2	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	KEY2	PBS0	AN	—	Touch key input 2
PB2/KEY3	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	KEY3	PBS0	AN	—	Touch key input 3
PB3/KEY4	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	KEY4	PBS0	AN	—	Touch key input 4
AN0~AN1	AN0~AN1	—	AN	—	A/D Converter external input channel
VOREG	VOREG	—	—	AN	LDO output pin
		—	AN	—	Positive power supply for ADC and PGA
VDD/AVDD	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
VSS/AVSS	VSS	—	PWR	—	Digital negative power supply, ground
	AVSS	—	PWR	—	Analog negative power supply, ground

Legend: I/T: Input type;

PWR: Power;

CMOS: CMOS output;

ST: Schmitt Trigger input;

O/T: Output type;

OPT: Optional by register option;

NMOS: NMOS output;

AN: Analog signal

Absolute Maximum Ratings

Supply Voltage	$V_{SS}-0.3V$ to $6.0V$
Input Voltage	$V_{SS}-0.3V$ to $V_{DD}+0.3V$
Storage Tempe	$-60^{\circ}C$ to $150^{\circ}C$
Operating Temperature	$-40^{\circ}C$ to $85^{\circ}C$
I_{OH} Total	$-80mA$
I_{OL} Total	$100mA$
Total Power Dissipation	$500mW$

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

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V _{DD}	Operating Voltage – HIRC	f _{sys} =8MHz	1.8	—	5.5	V
	Operating Voltage – LIRC	f _{sys} =32kHz	1.8	—	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	1.8V	f _{sys} =32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	1.8V	f _{sys} =8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	

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

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

Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V _{DD}	Conditions					
I _{STB}	SLEEP Mode	1.8V	WDT off	—	0.45	0.80	7.00	μA
		3V		—	0.45	0.90	8.00	
		5V		—	0.5	2.0	10.0	
	IDLE0 Mode – LIRC	1.8V	f _{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE1 Mode – HIRC	1.8V	f _{SUB} on, f _{sys} =8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	

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}	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%	
		1.8V~5.5V	25°C	-10%	8	+5 %	
			-40°C~85°C	-15%	8	+10%	

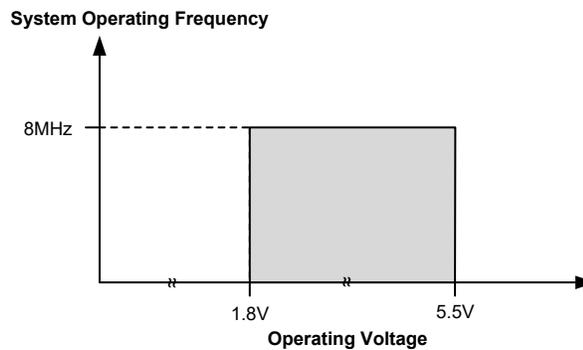
Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 1.8V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

Internal Low Speed Oscillator Characteristics – LIRC

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

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
t _{SST}	System Start-up Time (Wake-up from Condition where f _{sys} is off)	—	f _{sys} =f _H ~f _H /64, f _H =f _{HIRC}	—	16	—	t _{HIRC}
		—	f _{sys} =f _{SUB} =f _{LIRC}	—	2	—	t _{LIRC}
	System Start-up Time (Wake-up from Condition where f _{sys} is on)	—	f _{sys} =f _H ~f _H /64, f _H =f _{HIRC}	—	2	—	t _H
		—	f _{sys} =f _{SUB} =f _{LIRC}	—	2	—	t _{SUB}
	System Speed Switch Time (FAST to Slow Mode or SLOW to FAST Mode)	—	f _{HIRC} switches from off → on	—	16	—	t _{HIRC}
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 (LVR/WBTC Software Reset)	—	—				
	System Reset Delay Time (Reset Source from WDT Overflow Reset)	—	—	14	16	18	
t _{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

- Note: 1. For the System Start-up time values, whether f_{sys} is on or off depends upon the mode type and the chosen f_{sys} system oscillator. Details are provided in the System Operating Modes section.
2. The time units, shown by the symbols t_{HIRC}, t_{sys} etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example, t_{HIRC}=1/f_{HIRC}, t_{sys}=1/f_{sys} etc.
3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.
4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

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

For PA1, PA3~PA4, PA6 and PB0~PB3 pins.

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{IL}	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—	—	0	—	0.2V _{DD}	
V _{IH}	Input High Voltage for I/O Ports	5V	—	3.5	—	5.0	V
		—	—	0.8V _{DD}	—	V _{DD}	
I _{OL}	Sink Current for I/O Ports	3V	V _{OL} =0.1V _{DD}	16	32	—	mA
		5V		32	65	—	
I _{OH}	Source Current for I/O Ports	3V	V _{OH} =0.9V _{DD}	-4	-8	—	mA
		5V		-8	-16	—	
R _{PH}	Pull-high Resistance for I/O Ports ^(Note)	3V	—	20	60	100	kΩ
		5V		10	30	50	
I _{LEAK}	Input Leakage Current for I/O Ports	5V	V _{IN} =V _{DD} OR V _{IN} =V _{SS}	—	—	±1	μA
t _{INT}	External Interrupt Input Pin Minimum Pulse Width	—	—	10	—	—	μs
t _{TCK}	TM Clock Input Pin Minimum Pulse Width	—	—	0.3	—	—	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

For PA0, PA2 and PA5 pins.

 $T_a = -40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{DD}	Power Supply – V _{DD0}	—	—	1.8	5.0	5.5	V
V _{DDIO}	Power Supply – V _{DD1}	—	—	1.8	—	V _{DD}	V
V _{IL}	Input Low Voltage	5V	Pin power=V _{DDn} , V _{DDn} =V _{DD} , n=0~1	0	—	1.5	V
		—	Pin power=V _{DDn} , n=0~1	0	—	0.2V _{DDn}	
V _{IH}	Input High Voltage for I/O Ports	5V	Pin power=V _{DDn} , V _{DDn} =V _{DD} , n=0~1	3.5	—	5.0	V
		—	Pin power=V _{DDn} , n=0~1	0.8V _{DDn}	—	V _{DDn}	
I _{OL}	Sink Current for I/O Ports	3V	V _{OL} =0.1V _{DDn} , V _{DDn} =V _{DD} , n=0~1	16	32	—	mA
		5V	V _{OL} =0.1V _{DDn} , V _{DDn} =V _{DD} , n=0~1	32	65	—	
I _{OH}	Source Current for I/O Ports	3V	V _{OH} =0.9V _{DDn} , V _{DDn} =V _{DD} , n=0~1	-4	-8	—	mA
		5V	V _{OH} =0.9V _{DDn} , V _{DDn} =V _{DD} , n=0~1	-8	-16	—	
			V _{OH} =0.9V _{DDn} , V _{DDn} =3V, n=0~1	-2.5	-5.0	—	
R _{PH}	Pull-high Resistance for I/O Ports ⁽¹⁾	3V	LVPU=0, V _{DDn} =V _{DD} , n=0~1	20	60	100	kΩ
		5V	LVPU=0, V _{DDn} =V _{DD} , n=0~1	10	30	50	
			LVPU=0, V _{DDn} =3V, n=0~1	36	110	180	
		3V	LVPU=1, V _{DDn} =V _{DD} , n=0~1	6.67	15	23	
		5V	LVPU=1, V _{DDn} =V _{DD} , n=0~1	3.5	7.5	12	
LVPU=1, V _{DDn} =3V, n=0~1	9		27.5	45			
I _{LEAK}	Input Leakage Current for I/O Ports	5V	V _{IN} =V _{SS} or V _{IN} =V _{DDn} , n=0~1	—	—	±1	μA

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

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

Memory Electrical Characteristics
 $T_a = -40^{\circ}\text{C} \sim 85^{\circ}\text{C}$, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
Flash Program Memory							
V _{DD}	V _{DD} for Read/Write	—	—	1.8	—	5.5	V
t _{FWR}	Write Time	—	FWERTS=0	—	2.2	2.7	ms
			FWERTS=1	—	3.0	3.6	
t _{FER}	Erase Time	—	FWERTS=0	—	3.2	3.9	ms
			FWERTS=1	—	3.7	4.5	
E _p	Cell Endurance	—	—	100K	—	—	E/W
t _{RETD}	Data Retention Time	—	T _a =25°C	—	40	—	Year
t _{ACTV}	ROM Activation Time – Wake-up from Power Down Mode	—	—	32	—	64	μs
Data EEPROM Memory							
V _{DD}	V _{DD} for Read/Write	—	—	1.8	—	5.5	V
t _{EERD}	Read Time	—	—	—	—	4	t _{sys}

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
t _{EEWR}	Write Time(Byte Mode)	—	EWERTS=0	—	5.4	6.6	ms
			EWERTS=1	—	6.7	8.1	
	Write Time(Page Mode)		EWERTS=0	—	2.2	2.7	
			EWERTS=1	—	3.0	3.6	
t _{EEER}	Erase Time	—	EWERTS=0	—	3.2	3.9	ms
			EWERTS=1	—	3.7	4.5	
E _P	Cell Endurance	—	—	100K	—	—	E/W
t _{RETD}	Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V _{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

- Note: 1. The ROM activation time t_{ACTV} should be added when calculating the total system start-up time of a wake-up from the power down mode.
 2. “E/W” means Erase/Write times.

LVR/LVD Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit	
		V _{DD}	Conditions					
V _{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 1.7V	-5%	1.7	+5%	V	
			LVR enable, voltage select 1.9V		1.9			
			LVR enable, voltage select 2.55V	-3%	2.55	+3%		
			LVR enable, voltage select 3.15V		3.15			
			LVR enable, voltage select 3.8V		3.8			
V _{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 1.8V	-5%	1.8	+5%	V	
			LVD enable, voltage select 2.0V		2.0			
			LVD enable, voltage select 2.4V		2.4			
			LVD enable, voltage select 2.7V		2.7			
			LVD enable, voltage select 3.0V		3.0			
			LVD enable, voltage select 3.3V		3.3			
			LVD enable, voltage select 3.6V		3.6			
			LVD enable, voltage select 4.0V		4.0			
I _{LVR/LVD}	Operating Current	3V	LVD enable, LVR enable, V _{LVR} =1.9V, V _{LVD} =2V	—	—	10	μA	
		5V		—	10	15		
t _{LVDS}	LVDO Stable Time	—	For LVR enable, LVD off→on	—	—	18	μs	
			For LVR disable, LVD off→on	—	—	150		
t _{LVR}	Minimum Low Voltage Width to Reset	—	TLVR[1:0]=00B	120	240	480	μs	
			TLVR[1:0]=01B	0.5	1.0	2.0		
			TLVR[1:0]=10B	1	2	4		ms
			TLVR[1:0]=11B	2	4	8		
t _{LVD}	Minimum Low Voltage Width to Interrupt	—	TLVD[1:0]=00B/11B	50	80	140	μs	
			TLVD[1:0]=01B	80	140	260		
			TLVD[1:0]=10B	140	260	500		
I _{LVR}	Additional Current for LVR Enable	5V	LVD disable	—	—	14	μA	
I _{LVD}	Additional Current for LVD Enable	5V	LVR disable	—	—	14	μA	

24-Bit Delta Sigma A/D Converter Electrical Characteristics

$V_{DD}=V_{IN}$, $T_a=25^{\circ}\text{C}$, unless otherwise specified
LDO test conditions: MCU enters SLEEP mode, other functions disabled

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
LDO							
V_{IN}	LDO Input Voltage	—	LDOVS[1:0]=00B/01B	2.5	—	5.5	V
			LDOVS[1:0]=10B/11B	2.6	—	5.5	
I_Q	LDO Quiescent Current	—	LDOVS[1:0]=00B, $V_{IN}=3.6\text{V}$, no load	—	35	50	μA
I_{BGP}	Bandgap Quiescent Current	—	BGPEN=1, $V_{IN}=3.6\text{V}$, no load	—	150	200	μA
V_{OUT_LDO}	LDO Output Voltage	—	LDOVS[1:0]=00B, $V_{IN}=3.6\text{V}$, $I_{LOAD}=0.1\text{mA}$	-5%	1.8	+5%	V
			LDOVS[1:0]=01B, $V_{IN}=3.6\text{V}$, $I_{LOAD}=0.1\text{mA}$		2.2		
			LDOVS[1:0]=10B, $V_{IN}=3.6\text{V}$, $I_{LOAD}=0.1\text{mA}$		2.4		
			LDOVS[1:0]=11B, $V_{IN}=3.6\text{V}$, $I_{LOAD}=0.1\text{mA}$		3.0		
ΔV_{LOAD}	LDO Load Regulation ⁽¹⁾	—	LDOVS[1:0]=00B, $V_{IN}=V_{OUT_LDO}+0.7\text{V}$, $0\text{mA}\leq I_{LOAD}\leq 10\text{mA}$	—	0.105	0.210	%/mA
V_{DROPP_LDO}	LDO Dropout Voltage ⁽²⁾	—	LDOVS[1:0]=00B, $V_{IN}=3.6\text{V}$, $I_{LOAD}=10\text{mA}$, $\Delta V_{OUT_LDO}=2\%$	—	—	550	mV
			LDOVS[1:0]=01B, $V_{IN}=3.6\text{V}$, $I_{LOAD}=10\text{mA}$, $\Delta V_{OUT_LDO}=2\%$	—	—	250	mV
			LDOVS[1:0]=10B, $V_{IN}=3.6\text{V}$, $I_{LOAD}=10\text{mA}$, $\Delta V_{OUT_LDO}=2\%$	—	—	180	mV
			LDOVS[1:0]=11B, $V_{IN}=3.6\text{V}$, $I_{LOAD}=10\text{mA}$, $\Delta V_{OUT_LDO}=2\%$	—	—	160	mV
TC_{LDO}	LDO Temperature Coefficient	—	$T_a=-40^{\circ}\text{C}\sim 85^{\circ}\text{C}$, LDOVS[1:0]=01B, $V_{IN}=3.6\text{V}$, $I_{LOAD}=100\mu\text{A}$	—	—	200	ppm/ $^{\circ}\text{C}$
ΔV_{LINE_LDO}	LDO Line Regulation	—	LDOVS[1:0]=01B, $2.5\text{V}\leq V_{IN}\leq 5.5\text{V}$, $I_{LOAD}=100\mu\text{A}$	—	—	0.7	%/V
			LDOVS[1:0]=01B, $2.5\text{V}\leq V_{IN}\leq 3.6\text{V}$, $I_{LOAD}=100\mu\text{A}$	—	—	0.2	%/V
A/D Converter & A/D Converter Internal Reference Voltage (Delta Sigma A/D Converter)							
V_{ADC}	Supply Voltage for A/D Converter, PGA	—	LDOEN=0	2.4	—	3.0	V
			LDOEN=1	2.4	—	3.0	V
$I_{ADC+PGA}$	Additional Current for A/D Converter Enable	—	—	—	525	800	μA
I_{ADSTB}	Standby Current	—	MCU enters SLEEP mode, no load	—	—	1	μA
N_R	Resolution	—	—	—	—	24	Bit
INL	Integral Non-linearity	—	$V_{OREG}=3.0\text{V}$, $V_{REF}=V_{OREG}/2$, $\Delta SI=\pm 450\text{mV}$, PGA gain=1	—	± 50	—	ppm
NFB	Noise Free Bits	—	$f_{MCLK}=4\text{MHz}$, FLMS[1:0]=00B, $V_{OREG}=3.0\text{V}$, $V_{REF}=V_{OREG}/2$, PGA gain=128, OSR=16384	—	16	—	Bit
$ENOB$	Effective Number of Bits	—	$f_{MCLK}=4\text{MHz}$, FLMS[1:0]=00B, $V_{OREG}=3.0\text{V}$, $V_{REF}=V_{OREG}/2$, PGA gain=128, OSR=16384	—	18.7	—	Bit
f_{ADCK}	A/D Converter Clock Frequency	—	—	40.0	409.6	440.0	kHz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
f _{ADO}	A/D Converter Output Data Rate	—	f _{MCLK} =4MHz, FLMS[1:0]=00B, SINCS=0B	4.0	—	16.7	kHz
			f _{MCLK} =4MHz, FLMS[1:0]=10B, SINCS=0B	10.0	—	41.7	
V _{REFP}	Reference Input Voltage	—	—	V _{REFN} +0.8	—	V _{OREG}	V
V _{REFN}				0	—	V _{REFP} -0.8	V
V _{REF}				V _{REF} =V _{REFP} -V _{REFN}	0.80	—	1.75
PGA							
ΔD _i	Differential Input Voltage Range	—	Gain=PGAGN×ADGN	-V _{REF} /Gain	—	+V _{REF} /Gain	V
V _{CM_PGA}	Common Mode Voltage Range	—	—	0.4	—	V _{OREG} -0.95	V
Temperature Sensor							
TC _{TS}	Temperature Sensor Temperature Coefficient	—	T _a =-40°C~85°C	130	175	—	μV/°C
I _{TS}	Additional Current for Temperature Sensor Enable	—	—	—	100	120	μA
T _{ACC}	Temperature Accuracy (Linearity Error)	—	T _a =-10°C~50°C	-0.2	—	+0.2	°C

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

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

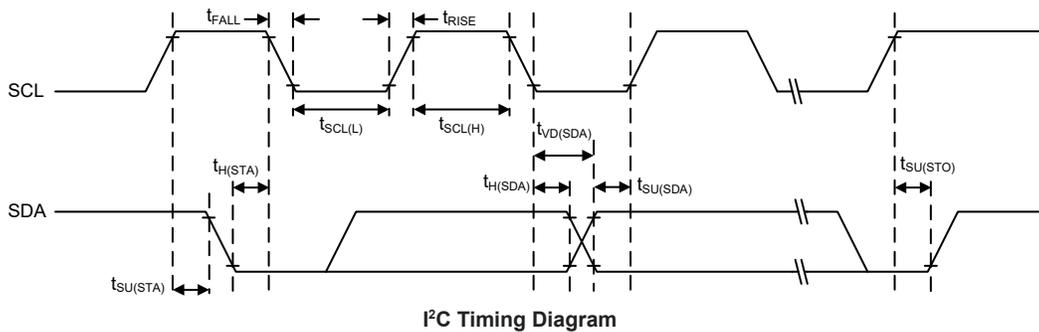
I²C Electrical Characteristics

T_a=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
f _{I2C}	I ² C Standard Mode (100kHz) f _{sys} Frequency ^(Note)	—	No debounce	2	—	—	MHz
			2 system clock debounce	4	—	—	
			4 system clock debounce	4	—	—	
	I ² C Fast Mode (400kHz) f _{sys} Frequency ^(Note)	—	No 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	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
t _{RISE}	SCL and SDA Rise Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
t _{SU(SDA)}	SDA Data Setup Time	3V/5V	Standard mode	0.25	—	—	μs
			Fast mode	0.1	—	—	
t _{H(SDA)}	SDA Data Hold Time	3V/5V	—	0.1	—	—	μs
t _{VD(SDA)}	SDA Data Valid Time	3V/5V	—	—	—	0.6	μs
t _{SU(STA)}	Start Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	
t _{H(STA)}	Start Condition Hold Time	3V/5V	Standard mode	4.0	—	—	μs
			Fast mode (f _{sys} ≥8MHz)	0.6	—	—	
			Fast mode (f _{sys} <8MHz)	0.8	—	—	
t _{SU(STO)}	Stop Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	

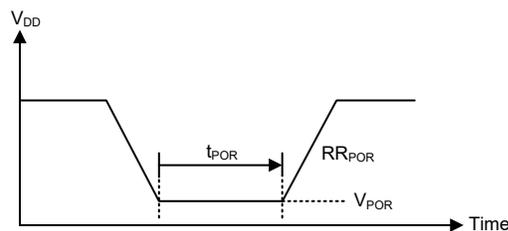
Note: Using the debounce function can make the transmission more stable and reduce the probability of communication failure due to interference.



Power-on Reset Characteristics

T_a=25°C

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

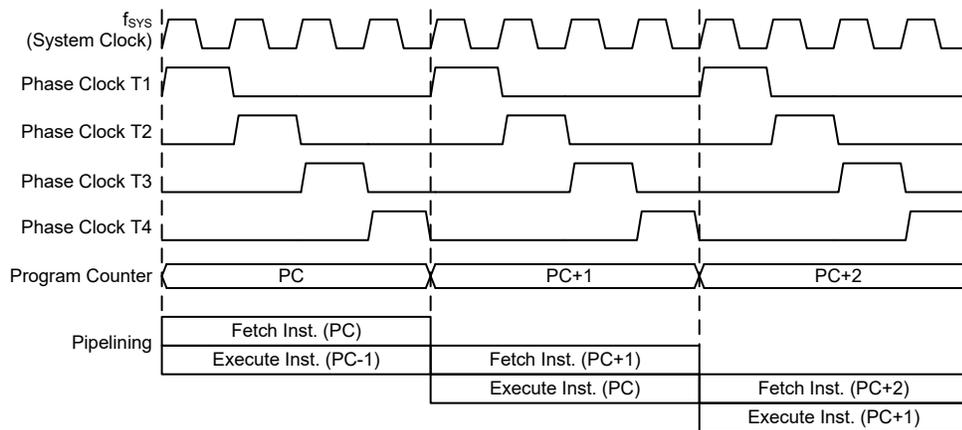


System Architecture

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

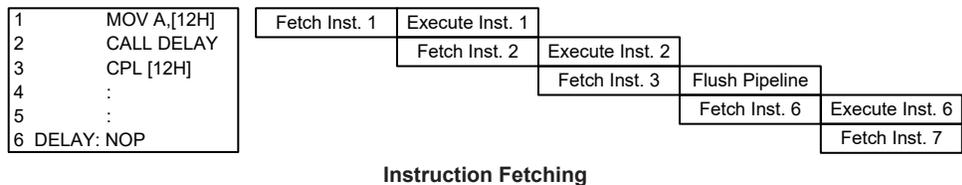
Clocking and Pipelining

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



System Clocking and Pipelining

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



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. 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)
PC12~PC8	PCL7~PCL0

Program Counter

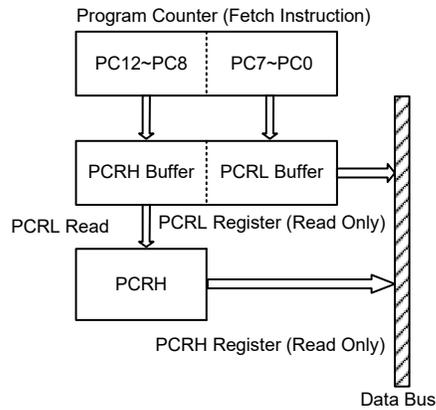
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Program Counter Read Registers

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

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

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



• **PCRL Register**

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

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

• **PCRH Register**

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

Bit 7~5 Unimplemented, read as “0”

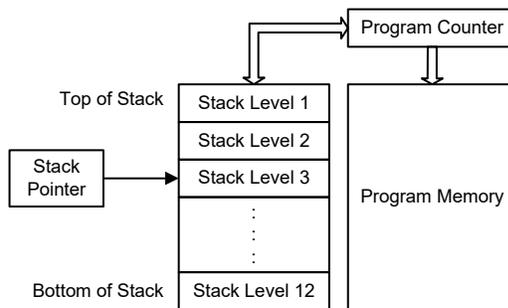
Bit 4~0 **D12~D8**: Program Counter Read High byte register bit 4 ~ bit 0

Stack

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

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

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



• **STKPTR Register**

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

Bit 7 **OSF**: Stack overflow flag
 0: No stack overflow occurred
 1: Stack overflow occurred

When the stack is full and a CALL instruction is executed or when the stack is empty and a RET instruction is executed, the OSF bit will be set high. The OSF bit is cleared only by software and cannot be reset automatically by hardware.

Bit 6~4 Unimplemented, read as “0”

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

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

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

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

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

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

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

Arithmetic and Logic Unit – ALU

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

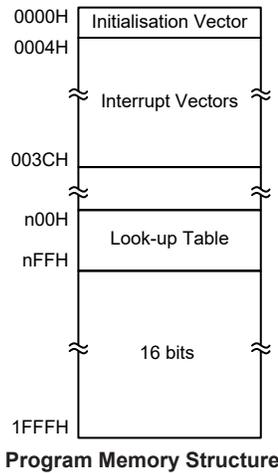
- Arithmetic operations:
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
LRR, LRRCA,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

Flash Program Memory

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

Structure

The Program Memory has a capacity of 8K×16 bits for the device. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer registers.



Special Vectors

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

Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the “TABRD [m]” or “TABRDL [m]” instructions respectively when the 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. Any unused bits in this transferred higher order byte will be read as “0”.

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

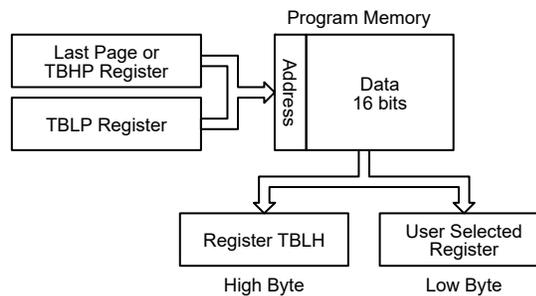


Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is “1F00H” which refers to the start address of the last page within the 8K words Program Memory of the device. The table pointer low byte register is setup here to have an initial value of “06H”. This will ensure that the first data read from the data table will be at the Program Memory address “1F06H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by the TBHP and TBLP registers 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

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

In Circuit Programming – ICP

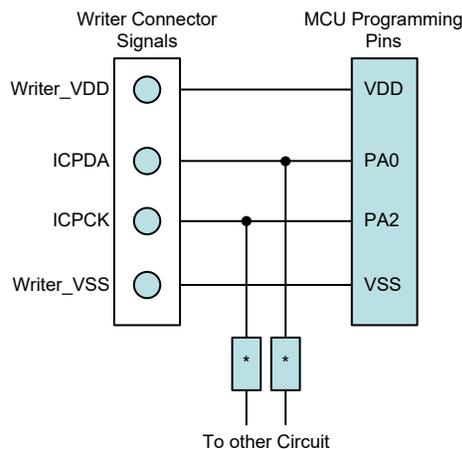
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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.

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

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

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



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

On-Chip Debug Support – OCDS

There is an EV chip named BS45V6052 which is used to emulate the BS45F6052 device. The EV chip device also provides an “On-Chip Debug” function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for “On-Chip Debug” function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip device for debugging, other pin functions which are shared with the OCSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used

as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named “Holtek e-Link for 8-bit MCU OCDS User’s Guide”.

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

In Application Programming – IAP

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

Flash Memory Read/Write Size

The Flash memory Erase and Write operations are carried out in a page format while the Read operation is carried out in a word format. The page size and write buffer size are both assigned with a capacity of 32 words. Note that the Erase operation should be executed before the Write operation is executed.

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

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

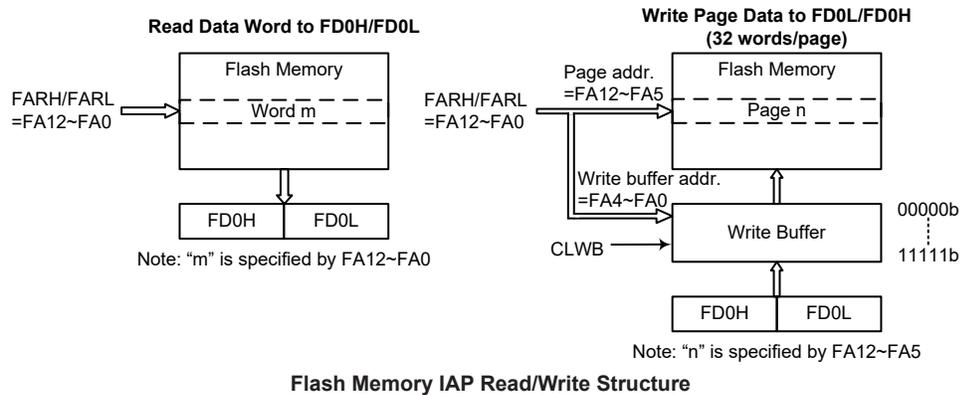
Operations	Format
Erase	32 words/page
Write	32 words/time
Read	1 word/time

Note: Page size=Write buffer size=32 words.

IAP Operation Format

Page	FARH	FARL[7:5]	FARL[4:0]
0	0000 0000	000	Tag Address
1	0000 0000	001	
2	0000 0000	010	
3	0000 0000	011	
4	0000 0000	100	
:	:	:	
:	:	:	
254	0001 1111	110	
255	0001 1111	111	

Page Number and Address Selection



Write Buffer

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

The write buffer size is 32 words corresponding to a page. The write buffer address is mapped to a specific Flash memory page specified by the memory address bits, FA12~FA5. The data written into the FD0L and FD0H registers will be loaded into the write buffer. When data is written into the high byte data register, FD0H, it will result in the data stored in the high and low byte data registers both being written into the write buffer. It will also cause the Flash memory address to be incremented by one, after which the new address will be loaded into the FARH and FARL address registers. When the Flash memory address reaches the page boundary, 11111b of a page with 32 words, the address will now not be incremented but stop at the last address of the page. At this point a new page address should be specified for any other erase/write operations.

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

IAP Flash Program Memory Registers

There are two address registers, four pairs of 16-bit data registers and three control registers, which are all located in Sector 0. Read and Write operations to the Flash memory are carried out using 16-bit data operations using the address and data registers and the control register. Several registers control the overall operation of the internal Flash Program Memory. The address registers are named FARL and FARH, the data registers are named FDnL and FDnH and the control registers are named FC0, FC1 and FC2.

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	—	—	—	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Register List

• **FC0 Register**

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

Bit 7 **CFWEN:** Flash Memory Erase/Write function enable control
 0: Flash memory erase/write function is disabled
 1: Flash memory erase/write function has been successfully enabled
 When this bit is cleared to 0 by application program, the Flash memory erase/write function is disabled. Note that this bit cannot be set high by application programs. Writing a “1” into this bit results in no action. This bit is used to indicate the Flash memory erase/write function status. When this bit is set to 1 by hardware, it means that the Flash memory erase/write function is enabled successfully. Otherwise, the Flash memory erase/write function is disabled if the bit is zero.

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

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

Bit 3 **FWPEN:** Flash memory Erase/Write function enable procedure trigger
 0: Erase/Write function enable procedure is not triggered or procedure timer times out
 1: Erase/Write function enable procedure is triggered and procedure timer starts to count
 This bit is used to activate the Flash memory Erase/Write function enable procedure and an internal timer. It is set by the application programs and then cleared by hardware when the internal timer times out. The correct patterns must be written into the FD1L/FD1H, FD2L/FD2H and FD3L/FD3H register pairs respectively as soon as possible after the FWPEN bit is set high.

- Bit 2 **FWT**: Flash memory write initiate control
 0: Do not initiate Flash memory write or indicating that a Flash memory write process has completed
 1: Initiate Flash memory write process
 This bit is set by software and cleared by hardware when the Flash memory write process has completed.
- Bit 1 **FRDEN**: Flash memory read enable control
 0: Flash memory read disable
 1: Flash memory read enable
 This is the Flash memory Read Enable Bit which must be set high before any Flash memory read operations are carried out. Clearing this bit to zero will inhibit Flash memory read operations.
- Bit 0 **FRD**: Flash memory read initiate control
 0: Do not initiate Flash memory read or indicating that a Flash memory read process has completed
 1: Initiate Flash memory read process
 This bit is set by software and cleared by hardware when the Flash memory read process has completed.

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

• **FC1 Register**

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

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

• **FC2 Register**

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

- Bit 7~2 Unimplemented, read as “0”
- Bit 1 **FWERTS**: Erase time and Write time selection
 0: Erase time is 3.2ms (t_{FER})/Write time is 2.2ms (t_{FWR})
 1: Erase time is 3.7ms (t_{FER})/Write time is 3.0ms (t_{FWR})
- Bit 0 **CLWB**: Flash memory Write Buffer Clear control
 0: Do not initiate a Write Buffer Clear process or indicating that a Write Buffer Clear process has completed
 1: Initiate Write Buffer Clear process
 This bit is set by software and cleared by hardware when the Write Buffer Clear process has completed.

• **FARL Register**

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

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

• **FARH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	FA12	FA11	FA10	FA9	FA8
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as “0”

Bit 4~0 **FA12~FA8**: Flash Memory Address bit 12 ~ bit 8

• **FD0L Register**

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

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

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

• **FD0H Register**

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

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

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

• **FD1L Register**

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

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

• **FD1H Register**

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

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

• **FD2L Register**

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

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

• **FD2H Register**

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

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

• **FD3L Register**

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

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

• **FD3H Register**

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

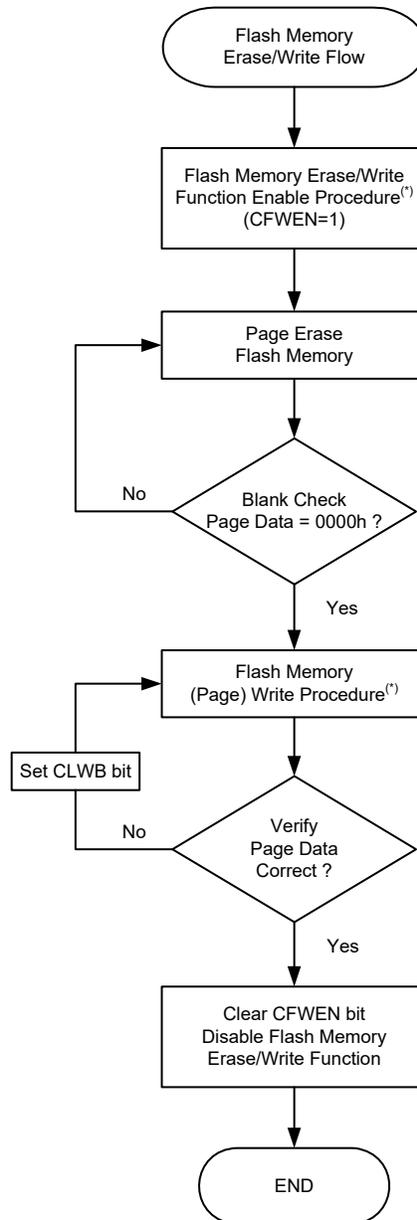
Bit 7~0 **D15~D8**: The fourth Flash Memory data word bit 15 ~ bit 8

Flash Memory Erase/Write Flow

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

Flash Memory Erase/Write Flow Descriptions

1. Activate the “Flash Memory Erase/Write function enable procedure” first. When the Flash Memory Erase/Write function is successfully enabled, the CFWEN bit in the FC0 register will automatically be set high by hardware. After this, Erase or Write operations can be executed on the Flash memory. Refer to the “Flash Memory Erase/Write Function Enable Procedure” for details.
2. Configure the Flash memory address to select the desired erase page, tag address and then erase this page. For a page erase operation, set the FARL and FARH registers to specify the start address of the erase page, then write dummy data into the FD0H register to tag address. The current address will be internally incremented by one after each dummy data is written into the FD0H register. When the address reaches the page boundary, 11111b, the address will not be further incremented but stop at the last address of the page. Note that the write operation to the FD0H register is used to tag address, it must be implemented to determine which addresses to be erased.
3. Execute a Blank Check operation to ensure whether the page erase operation is successful or not. The “TABRD” instruction should be executed to read the Flash memory contents and to check if the contents is 0000h or not. If the Flash memory page erase operation fails, users should go back to Step 2 and execute the page erase operation again.
4. Write data into the specific page. Refer to the “Flash Memory Write Procedure” for details.
5. Execute the “TABRD” instruction to read the Flash memory contents and check if the written data is correct or not. If the data read from the Flash memory is different from the written data, it means that the page write operation has failed. The CLWB bit should be set high to clear the write buffer and then write the data into the specific page again if the write operation has failed.
6. Clear the CFWEN bit to disable the Flash Memory Erase/Write function enable mode if the current page Erase and Write operations are complete if no more pages need to be erased or written.



Flash Memory Erase/Write Flow

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

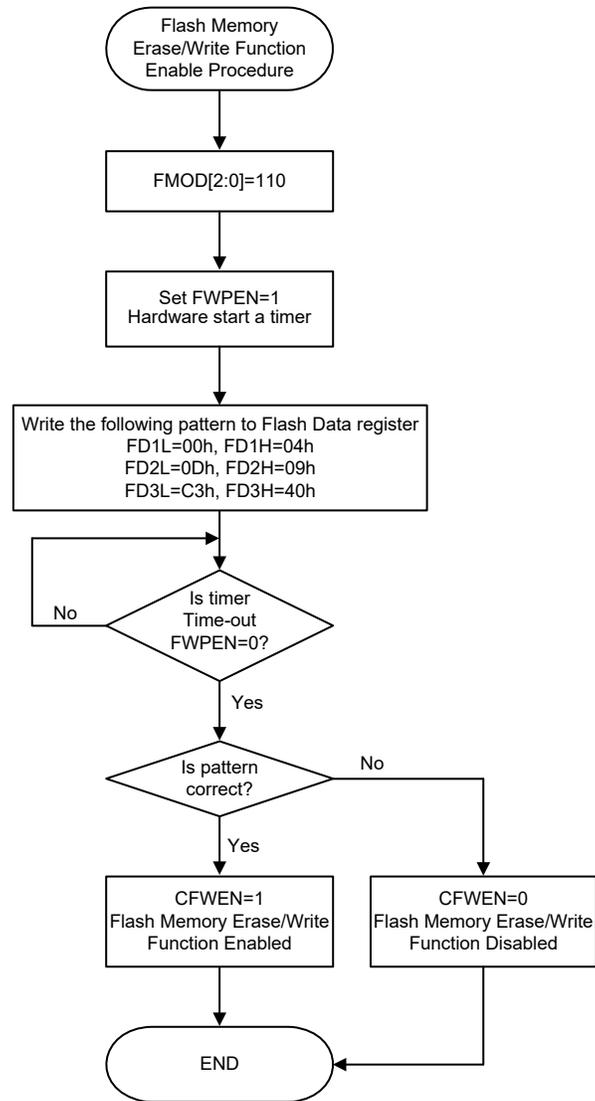
Flash Memory Erase/Write Function Enable Procedure

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

Flash Memory Erase/Write Function Enable Procedure Description

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

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



Flash Memory Erase/Write Function Enable Procedure

Flash Memory Write Procedure

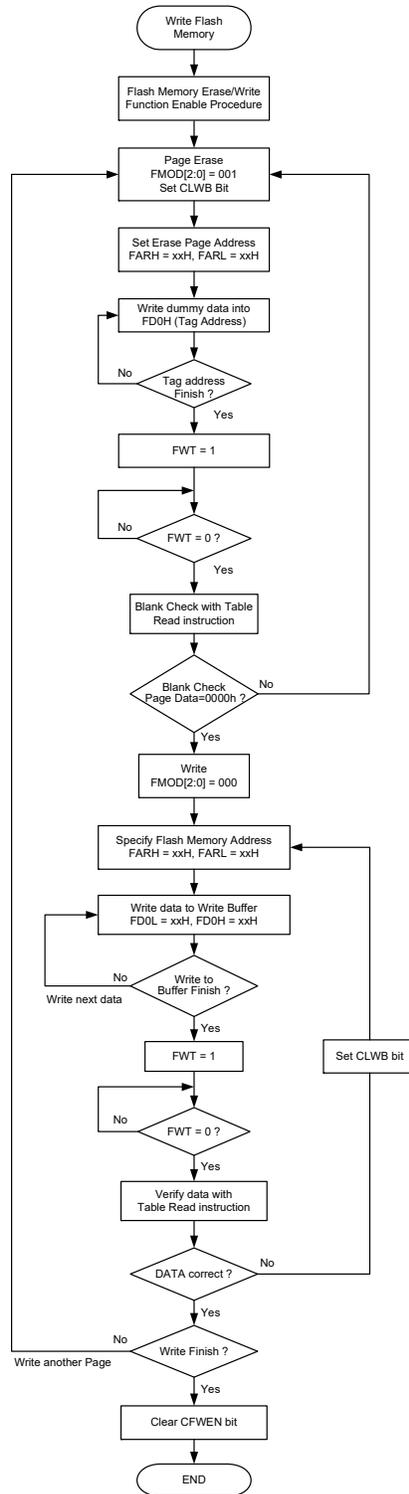
After the Flash memory erase/write function has been successfully enabled as the CFWEN bit is set high, the data to be written into the Flash memory can be loaded into the write buffer. The selected Flash memory page data should be erased by properly configuring the IAP control registers before the data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

The write buffer size is 32 words, known as a page, whose address is mapped to a specific Flash memory page specified by the memory address bits, FA12~FA5. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits specify.

Flash Memory Consecutive Write Description

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

1. Activate the “Flash Memory Erase/Write function enable procedure”. Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the “Flash Memory Erase/Write function enable procedure” for more details.
2. Set the FMOD field to “001” to select the erase operation and set the CLWB bit high to clear the write buffer. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers and has been tagged address. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.
Go to step 2 if the erase operation is not successful.
Go to step 4 if the erase operation is successful.
4. Set the FMOD field to “000” to select the write operation.
5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L and FD0H registers within a page as specified by their consecutive addresses. The maximum written data number is 32 words.
6. Set the FWT bit high to write the data words from the write buffer to the Flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed. If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 5.
Go to step 8 if the write operation is successful.
8. Clear the CFWEN bit low to disable the Flash memory erase/write function.



Flash Memory Consecutive Write Procedure

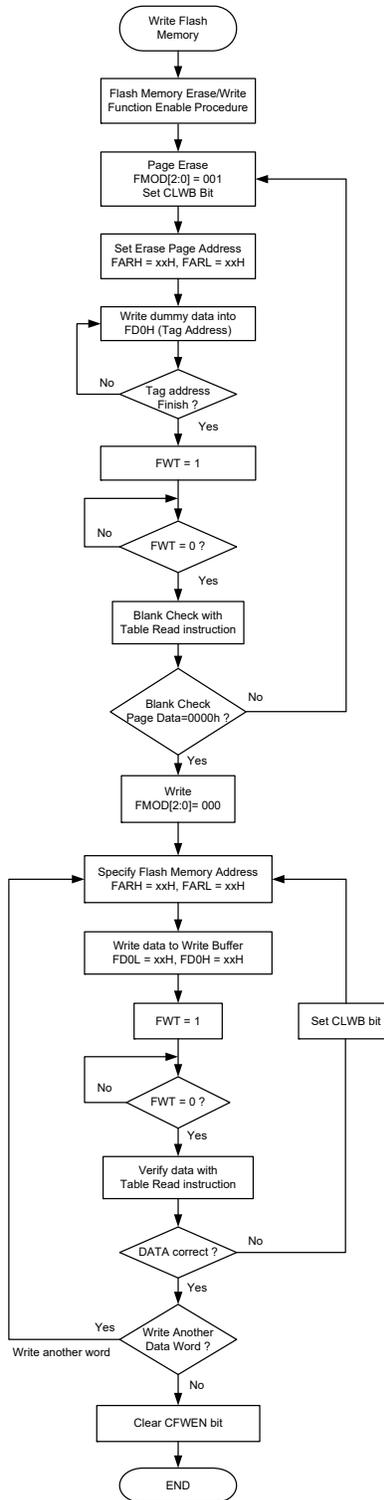
- Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.
 2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

Flash Memory Non-consecutive Write Description

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

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

1. Activate the “Flash Memory Erase/Write function enable procedure”. Check the CFWEN bit value and then execute the erase/write operation if the CFWEN bit is set high. Refer to the “Flash Memory Erase/Write function enable procedure” for more details.
2. Set the FMOD field to “001” to select the erase operation and set the CLWB bit high to clear the write buffer. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers and has been tagged address. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.
Go to step 2 if the erase operation is not successful.
Go to step 4 if the erase operation is successful.
4. Set the FMOD field to “000” to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
6. Set the FWT bit high to transfer the data word from the write buffer to the Flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed. If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 5.
Go to step 8 if the write operation is successful.
8. Setup the desired address ADDR2 in the FARH and FARL registers. Write the desired data word DATA2 first into the FD0L register and then into the FD0H register.
9. Set the FWT bit high to transfer the data word from the write buffer to the Flash memory. Wait until the FWT bit goes low.
10. Verify the data using the table read instruction to ensure that the write operation has successfully completed. If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 8.
Go to step 11 if the write operation is successful.
11. Clear the CFWEN bit low to disable the Flash memory erase/write function.



Flash Memory Non-consecutive Write Procedure

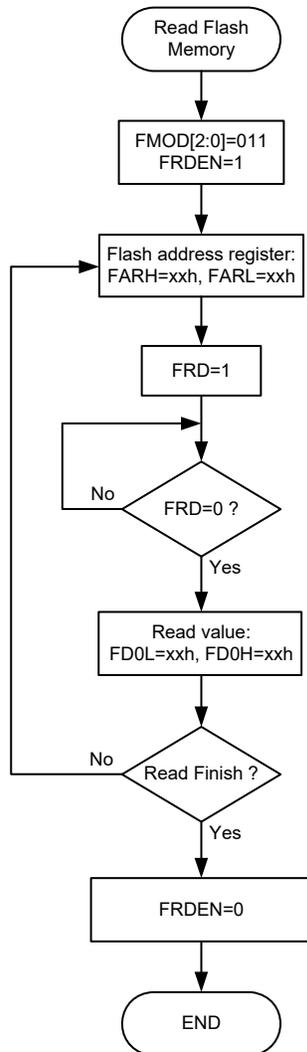
- Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.
 2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

Important Points to Note for Flash Memory Write Operations

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

Flash Memory Read Procedure

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



Flash Memory Read Procedure

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

Data Memory

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

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

The device also provides dedicated memory areas for the Touch Key data storage. In this chapter, only the General Purpose Data Memory and the Special Function Register Data Memory are introduced. More information about the Touch Key Data Memory can be obtained in its corresponding chapter.

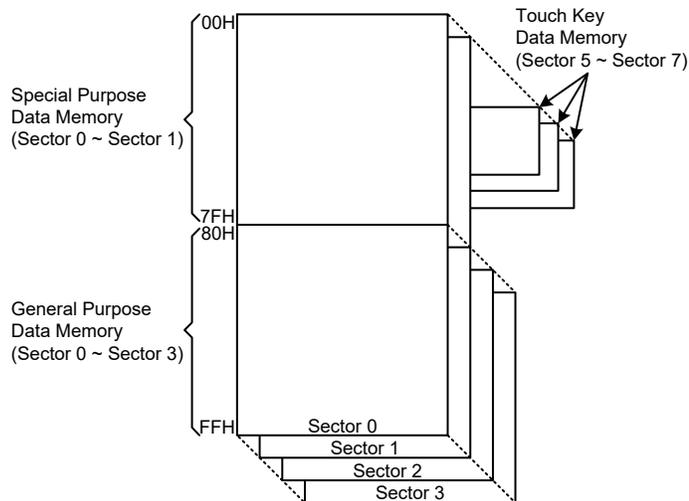
Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide RAM. Each of the Data Memory sector is categorised into two types, the special Purpose Data Memory and the General Purpose Data Memory. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH. The Touch Key Data Memory is located in Sector 5 ~ Sector 7 with a start address of 00H.

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

Special Purpose Data Memory	General Purpose Data Memory		Touch Key Data Memory	
Located Sectors	Capacity	Sector: Address	Capacity	Sector: Address
0, 1	512×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH	24×8	5: 00H~07H 6: 00H~07H 7: 00H~07H

Data Memory Summary



Data Memory Structure

Data Memory Addressing

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

Direct Addressing can be used in all sectors using the extended instruction which can address all available data memory space. For the accessed data memory which is located in any data memory sector 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 10 valid bits for this device, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0	Sector 1		Sector 0	Sector 1
00H	IAR0	TMR	40H		EEC
01H	MP0	TMRC	41H	EEAL	
02H	IAR1	PAS0	42H	EEAH	
03H	MP1L	PAS1	43H	EED	
04H	MP1H	PBS0	44H	TKTMR	
05H	ACC	PMPS	45H	TKC0	
06H	PCL	TLVRC	46H	TKC1	
07H	TBLP	IFS	47H	TKC2	
08H	TBLH		48H	TK16DL	
09H	TBHP		49H	TK16DH	
0AH	STATUS		4AH	TKM0C0	
0BH			4BH	TKM0C1	
0CH	IAR2		4CH	TKM0C2	
0DH	MP2L		4DH	TKM016DL	
0EH	MP2H		4EH	TKM016DH	
0FH	RSTFC		4FH	TKM0ROL	
10H	INTC0		50H	TKM0ROH	
11H	INTC1		51H	TKM0TH16L	
12H	INTC2		52H	TKM0TH16H	
13H	INTC3		53H	TKM0THS	
14H	PA		54H	PWRC	
15H	PAC		55H	PGAC0	
16H	PAPU		56H	PGAC1	
17H	PAWU		57H	PGACS	
18H	PB		58H	ADRL	
19H	PBC		59H	ADRM	
1AH	PBPU		5AH	ADRH	
1BH	TB0C		5BH	ADCR0	
1CH	TB1C		5CH	ADCR1	
1DH	LVRC		5DH		
1EH	LVPUC		5EH		
1FH	PSC0R		5FH		
20H	PSC1R		60H	ADCS	
21H	SCC		61H		
22H	LVDC		62H		
23H	WDTC		63H		
24H	HIRCC		64H		
25H	MFI		65H		
26H	USR		66H		
27H	UCR1		67H		
28H	UCR2		68H		
29H	UCR3		69H	STKPTR	
2AH	BRDH		6AH	FC0	
2BH	BRDL		6BH	FC1	
2CH	UFCR		6CH	FC2	
2DH	TXR_RXR		6DH	FARL	
2EH	RxCNT		6EH	FARH	
2FH	CRCCR		6FH	FD0L	
30H	CRCIN		70H	FD0H	
31H	CRCDL		71H	FD1L	
32H	CRCDH		72H	FD1H	
33H	CTM0C0		73H	FD2L	
34H	CTM0C1		74H	FD2H	
35H	CTM0DL		75H	FD3L	
36H	CTM0DH		76H	FD3H	
37H	CTM0AL		77H	IICC0	
38H	CTM0AH		78H	IICC1	
39H	CTM1C0		79H	IICD	
3AH	CTM1C1		7AH	IICA	
3BH	CTM1DL		7BH	IICTOC	
3CH	CTM1DH		7CH	IECC	
3DH	CTM1AL		7DH	PCRL	
3EH	CTM1AH		7EH	PCRH	
3FH	INTEG		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 sections, however several registers require a separate description in this section.

Indirect Addressing Registers – IAR0, IAR1, IAR2

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

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

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L and MP2H, are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the 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.

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

Indirect Addressing Program Example

Example 1

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

Example 2

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block db ?
code .section at 0 'code'
org 00h
start:
    mov a,04h                ; setup size of block
    mov block,a
    mov a,01h                ; setup the memory sector
    mov mplh, 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.

Accumulator – ACC

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

Program Counter Low Byte Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Option Memory Mapping Register – ORMC

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

To successfully enable the Option Memory Mapping function, the specific pattern of 55H and AAH must be written into the ORMC register in two consecutive instruction cycles. It is therefore recommended that the global interrupt bit EMI should first be cleared before writing the specific pattern, and then set high again at a proper time according to users’ requirements after the pattern is successfully written. An internal timer will be activated when the pattern is successfully written. The mapping operation will be automatically finished after a period of $4 \times t_{LIRC}$. Therefore, users should read the data in time, otherwise the Option Memory Mapping function needs to be restarted. After the completion of each consecutive write operation to the ORMC register, the timer will recount.

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

• ORMC Register

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

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

Status Register – STATUS

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

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

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

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

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

• STATUS Register

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

“x”: unknown

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

Bit 6 **CZ:** The operational result of different flags for different instructions
 For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.
 For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the “AND” operation result which is performed by the previous operation CZ flag and current operation zero flag. For other instructions, the CZ flag will not be affected.

Bit 5	<p>TO: Watchdog Time-out flag 0: After power up or executing the “CLR WDT” or “HALT” instruction 1: A watchdog time-out occurred</p>
Bit 4	<p>PDF: Power down flag 0: After power up or executing the “CLR WDT” instruction 1: By executing the “HALT” instruction</p>
Bit 3	<p>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</p>
Bit 2	<p>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</p>
Bit 1	<p>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</p>
Bit 0	<p>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</p> <p>The “C” flag is also affected by a rotate through carry instruction.</p>

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 512×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and write operations to the EEPROM are carried out in either the byte mode or page mode determined by the mode selection bit, MODE, in the control register, EEC.

EEPROM Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEAL	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH	—	—	—	—	—	—	—	EEAH0
EED	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
EEC	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD

EEPROM Register List

• **EEAL Register**

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

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

• **EEAH Register**

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

Bit 7~1 Unimplemented, read as “0”
 Bit 0 **EAH0**: Data EEPROM address high byte register bit 0
 Data EEPROM address bit 8

• **EED Register**

Bit	7	6	5	4	3	2	1	0
Name	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
R/W								
POR	0	0	0	0	0	0	0	0

Bit 7~0 **EED7~EED0**: Data EEPROM data bit 7 ~ bit 0

• **EEC Register**

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

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

Bit 6 **EREN**: Data EEPROM erase enable
 0: Disable
 1: Enable

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

- Bit 5 **ER**: Data EEPROM erase control
 0: Erase cycle has finished
 1: Activate an erase cycle
 This is the Data EEPROM Erase Control Bit. When this bit is set high by the application program, an erase cycle will be activated. This bit will be automatically reset to zero by hardware after the erase cycle has finished. Setting this bit high will have no effect if the EREN has not first been set high.
- Bit 4 **MODE**: Data EEPROM operation mode selection
 0: Byte operation mode
 1: Page operation mode
 This is the EEPROM operation mode selection bit. When the bit is set high by the application program, the Page write, erase or read function will be selected. Otherwise, the byte write or read function will be selected. The EEPROM page buffer size is 16 bytes.
- Bit 3 **WREN**: Data EEPROM write enable
 0: Disable
 1: Enable
 This is the Data EEPROM Write Enable Bit, which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations. Note that the WREN bit will automatically be cleared to zero after the write operation is finished.
- Bit 2 **WR**: Data EEPROM write control
 0: Write cycle has finished
 1: Activate a write cycle
 This is the Data EEPROM Write Control Bit. When this bit is set high by the application program, a write cycle will be activated. This bit will be automatically reset to zero by hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.
- Bit 1 **RDEN**: Data EEPROM read enable
 0: Disable
 1: Enable
 This is the Data EEPROM Read Enable Bit, which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.
- Bit 0 **RD**: Data EEPROM read control
 0: Read cycle has finished
 1: Activate a read cycle
 This is the Data EEPROM Read Control Bit. When this bit is set high by the application program, a read cycle will be activated. This bit will be automatically reset to zero by hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

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

Read Operation from the EEPROM

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

Byte Read Mode

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

Page Read Mode

The EEPROM page read operation can be executed when the mode selection bit, MODE, is set high. The page size can be up to 16 bytes for the page read operation. For a page read operation the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM page read operation. Note that setting the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the current byte read cycle terminates, the RD bit will automatically be cleared to zero indicating that the EEPROM data can be read from the EED register and then the current address will be incremented by one by hardware. The data which is stored in the next EEPROM address can continuously be read when the RD bit is again set high without reconfiguring the EEPROM address and RDEN control bit. The application program can poll the RD bit to determine when the data is valid for reading.

The EEPROM address higher 5 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 5-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 5 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 5-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”.

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

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

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

Write Operation to the EEPROM

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

Byte Write Mode

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

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

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

Page Write Mode

Before a page write operation is executed, it is important to ensure that a relevant page erase operation has been successfully executed. The EEPROM page write operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page write. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM write enable control bit, namely WREN, is changed from “1” to “0”, the internal page buffer will also be cleared. Note that when the WREN bit is changed from “0” to “1”, the internal page buffer will not be cleared. A page write is initiated in the same way as a byte write initiation except that the EEPROM data can be written up to 16 bytes. The EEPROM address higher 5 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 5-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”. At this point any data write operations to the EED register will be invalid.

For page write operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers, then the data to be written should be placed in the EED register. The maximum data length for a page is 16 bytes. Note that when a data byte is written into the EED register, then the data in the EED register will be loaded into the internal page buffer and the current address value will automatically be incremented by one. When the page data is completely written into the page buffer, then the WREN bit in the EEC register should be set high to enable write operations and the WR bit must be immediately set high to initiate the EEPROM write process. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

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

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

Write Protection

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

EEPROM Interrupt

The EEPROM erase or write interrupt is generated when an EEPROM erase or write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. When an EEPROM erase or write cycle ends, the DEF request flag will be set. If the global and EEPROM interrupts are enabled and the stack is not full, a jump to the EEPROM Interrupt vector will take place. When the interrupt is serviced, the EEPROM interrupt flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. More details can be obtained in the Interrupts section.

Programming Considerations

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

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

Programming Examples

Reading a Data Byte from the EEPROM – polling method

```

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

```

Reading a Data Page from the EEPROM – polling method

```

MOV A, 40H                ; setup memory pointer low byte MP1L
MOV MP1L, A              ; MP1 points to EEC register
MOV A, 01H               ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4               ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H   ; user defined high byte address

```

```
MOV EEAH, A
MOV A, EEPROM_ADRES_L      ; user defined low byte address
MOV EEAL, A
SET IAR1.1                 ; set RDEN bit, enable read operations
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL READ
CALL READ
:
:
JMP PAGE_READ_FINISH
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
READ:
SET IAR1.0                 ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                  ; check for read cycle end
JMP BACK
MOV A, EED                  ; move read data to register
MOV READ_DATA, A
RET
:
PAGE_READ_FINISH:
CLR IAR1                    ; disable EEPROM read function
CLR MP1H
```

Erasing a Data Page to the EEPROM – polling method

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

SET EMI
BACK:
SZ IAR1.5                  ; check for erase cycle end
JMP BACK
CLR MP1H
```

Writing a Data Byte to the EEPROM – polling method

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

SET EMI
BACK:
SZ IAR1.2 ; check for write cycle end
JMP BACK
CLR MP1H
```

Writing a Data Page to the EEPROM – polling method

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

SET EMI
BACK:
SZ IAR1.2 ; check for write cycle end
JMP BACK
CLR MP1H
```

Oscillators

Various oscillator 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 the application program by using relevant control registers.

Oscillator Overview

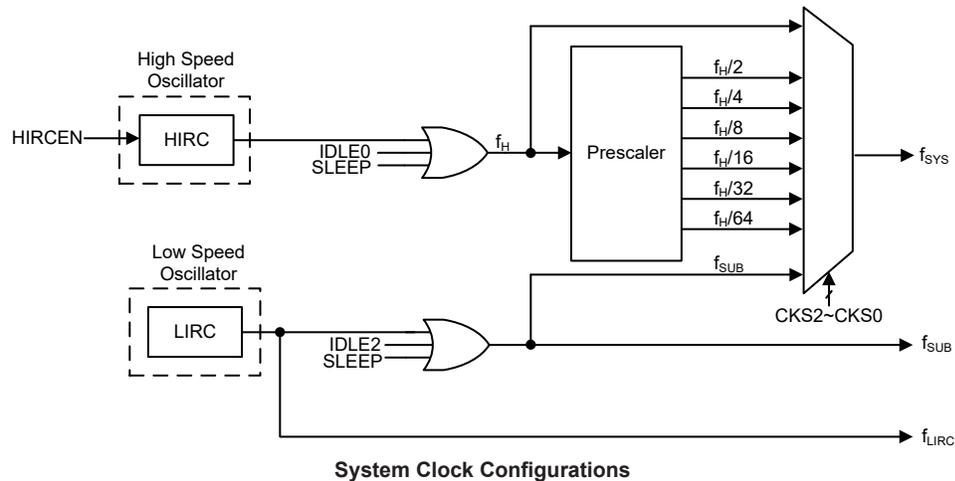
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators.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

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

Oscillator Types

System Clock Configurations

There are two methods of generating the system clock, a high speed oscillator and a low speed oscillator. The high speed oscillator is the internal 8MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.



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 a fixed frequency of 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is 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.

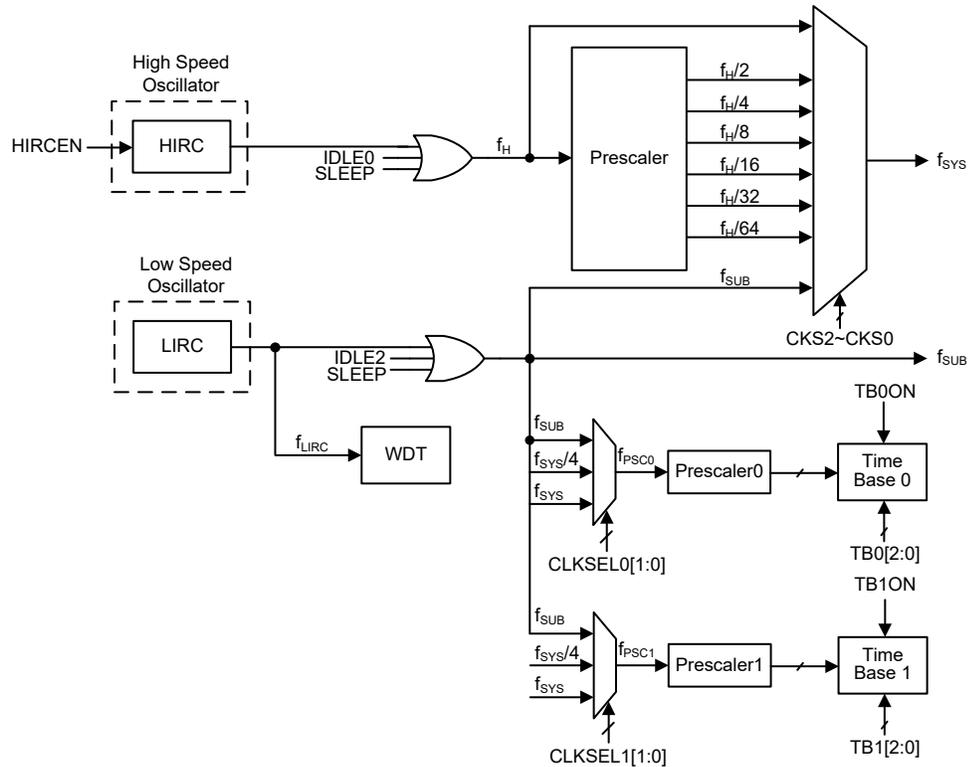
Operating Modes and System Clocks

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. The low speed system clock source is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$ ~ $f_H/64$.



Device Clock Configurations

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

System Operation Modes

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

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

"x": Don't care

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

2. The f_{LIRC} clock can be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.
3. The f_{LIRC} clock can be switched on by setting the M0TSS bit (touch key time slot counter clock source selection) high in the SLEEP mode.

FAST Mode

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

SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bits are low. In the SLEEP mode the CPU will be stopped, and the f_{SUB} clock to peripheral will be stopped too. However the f_{LIRC} clock will continue to operate if the WDT function is enabled by the WDTC register or if the touch key M0TSS bit is set high to select the f_{LIRC} clock source.

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

The registers, SCC and HIRCC, 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	—	—	—	FHIDEN	FSIDEN
HIRCC	—	—	—	—	—	—	HIRCF	HIRCEN

System Operating Mode Control Register List

• **SCC Register**

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	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 4~2 Unimplemented, read as “0”

Bit 1 **FHIDEN**: High Frequency oscillator control when CPU is switched off

- 0: Disable
- 1: Enable

This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing a “HALT” instruction.

Bit 0 **FSIDEN**: Low Frequency oscillator control when CPU is switched off

- 0: Disable
- 1: Enable

This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing a “HALT” instruction.

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

• **HIRCC Register**

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

Bit 7~2 Unimplemented, read as “0”

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

- 0: HIRC unstable
- 1: HIRC stable

This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set high to enable the HIRC oscillator, the HIRCF bit will first be cleared to zero and then set high after the HIRC oscillator is stable.

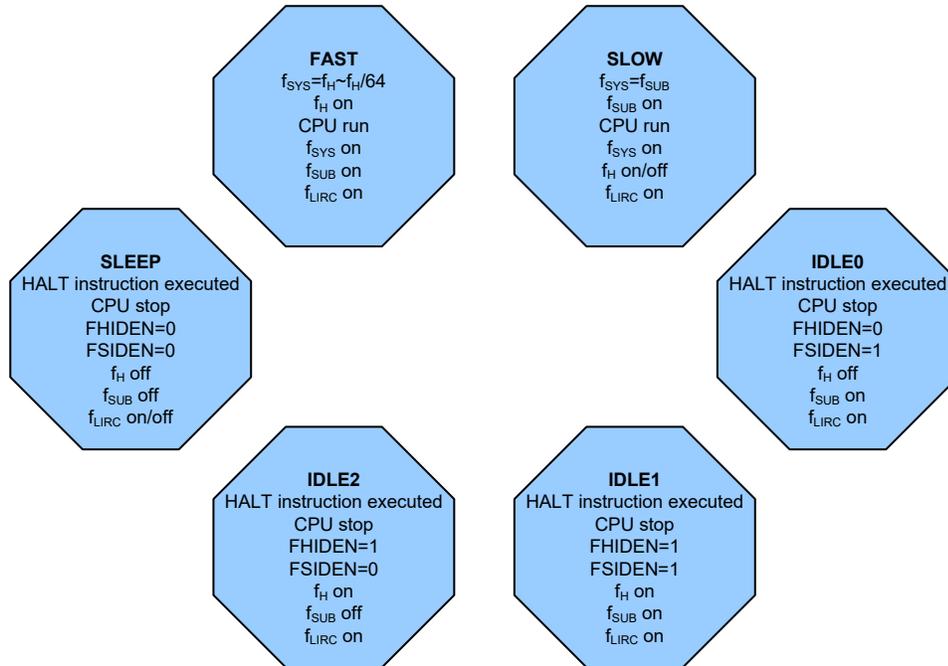
Bit 0 **HIRCEN**: HIRC oscillator enable control

- 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.

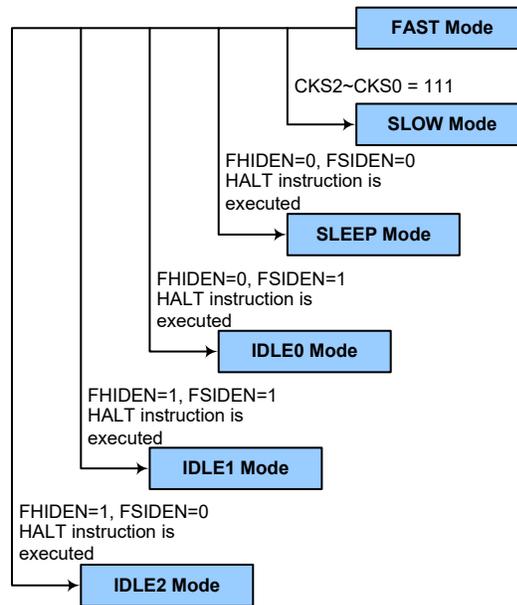
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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

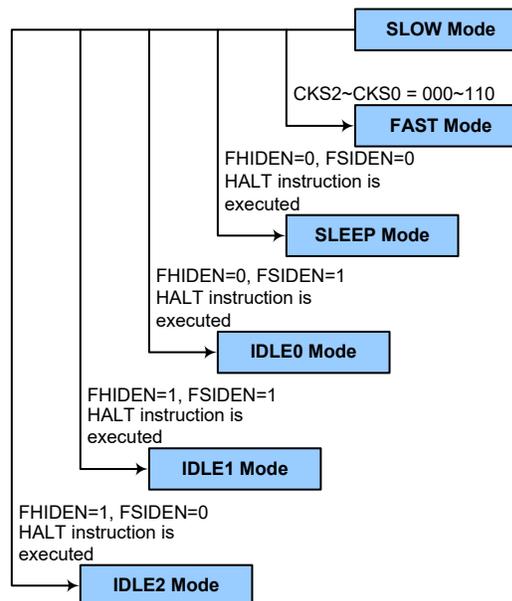
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

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

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



Entering the SLEEP Mode

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

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

Standby Current Considerations

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

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

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

Wake-up

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

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

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

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

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

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{LIRC} which is 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 timeout period as well as the WDT enable/disable and software reset MCU operation. This register controls the overall operation of the Watchdog Timer.

• 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: Reset MCU

If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET} , and the WRF bit in the RSTFC register will be set to 1.

Bit 2~0 **WS2~WS0:** WDT time-out period t_{WDT} 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	—	—	—	—	—	LVRF	LRF	WRF
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	x	0	0

“x”: unknown

- Bit 7~3 Unimplemented, read as “0”
- Bit 2 **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.
- Bit 1 **LRF**: LVR control register software reset flag
Refer to the Low Voltage Reset section.
- Bit 0 **WRF**: WDT control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the WDT control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

Watchdog Timer Operation

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

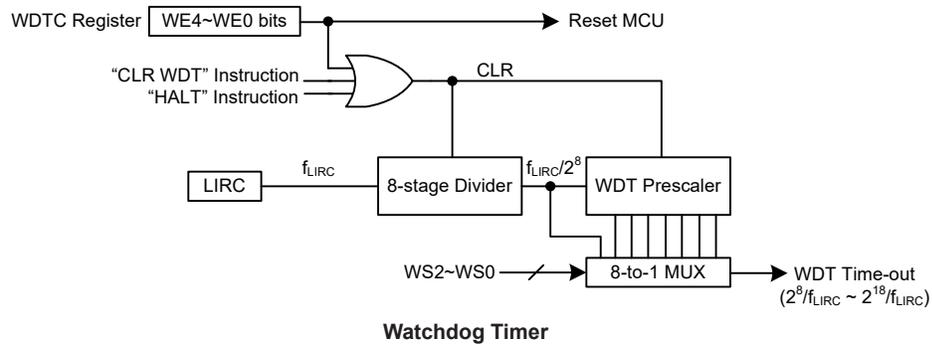
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

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 WDT software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 field, the second is using the Watchdog Timer software clear instruction and the third is via a HALT instruction.

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

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



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.

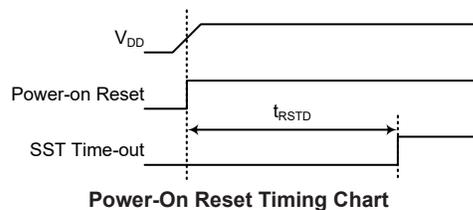
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

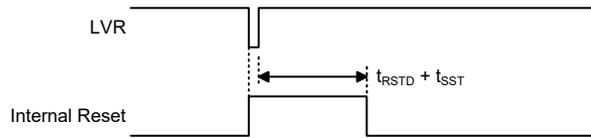
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. 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.



Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provide an MCU reset should the value fall below a certain predefined level. The LVR function can be enabled or disabled by the LVRC control register. If the supply voltage of the device drops to within a range of $0.9V \sim V_{LVR}$ such as might occur when changing the battery, the LVR will

automatically reset the device internally and the LVRF bit in the 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 $0.9V \sim V_{LVR}$ must exist for a time greater than that specified by t_{LVR} in the LVR/LVD Electrical Characteristics. If the duration of the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual t_{LVR} value can be selected by the TLVR1~TLVR0 bits in the TLVRC register. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits have any other values, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the device after a delay time, t_{SRESET} . When this happens, the LRF bit in the RSTFC register will be set to 1. Note that the LVR function will be automatically disabled when the device enters the SLEEP/IDLE mode.



Low Voltage Reset Timing Chart

• **LVRC Register**

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

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

- 01100110: 1.7V
- 01010101: 1.9V
- 00110011: 2.55V
- 10011001: 3.15V
- 10101010: 3.8V
- 11110000: LVR disable

Other values: Generates a MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the five defined LVR voltage values above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps for greater 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 six 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.

• **TLVRC Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **TLVR1~TLVR0: Minimum low voltage width to reset time (t_{LVR}) selection**

- 00: $(7 \sim 8) \times t_{LIRC}$
- 01: $(31 \sim 32) \times t_{LIRC}$
- 10: $(63 \sim 64) \times t_{LIRC}$
- 11: $(127 \sim 128) \times t_{LIRC}$

• **RSTFC Register**

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

“x”: unknown

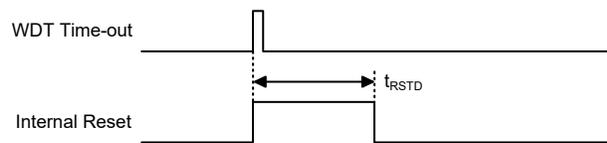
- Bit 7~3 Unimplemented, read as “0”
- Bit 2 **LVRF**: LVR function reset flag
 0: Not occurred
 1: Occurred
 This bit is set high when a specific Low Voltage Reset situation occurs. This bit can only be cleared to zero by the application program.
- Bit 1 **LRF**: LVR control register software reset flag
 0: Not occurred
 1: Occurred
 This bit is set high if the LVRC register contains any non-defined register values. This in effect acts like a software-reset function. This bit can only be cleared to zero by the application program.
- Bit 0 **WRF**: WDT control register software reset flag
 Refer to the Watchdog Timer Control Register section.

IAP Reset

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

Watchdog Time-out Reset during Normal Operation

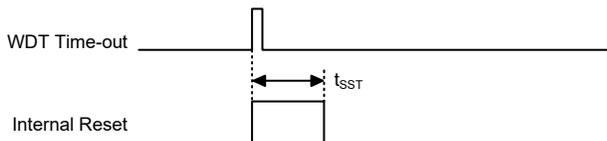
The Watchdog time-out Reset during normal operation in the FAST or SLOW Mode is the same as the hardware Low Voltage Reset except that the Watchdog time-out flag TO will be set to “1”.



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



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

Reset Initial Conditions

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

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

“u”: unchanged

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

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

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

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	---x xxxx	---u uuuu	---u uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	0000 0000	0000 0000	uuuu uuuu
PA	-111 1111	-111 1111	-uuu uuuu
PAC	-111 1111	-111 1111	-uuu uuuu
PAPU	-000 0000	-000 0000	-uuu uuuu

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PAWU	-000 0000	-000 0000	-uuu uuuu
PB	---- 1111	---- 1111	---- uuuu
PBC	---- 1111	---- 1111	---- uuuu
PBPU	---- 0000	---- 0000	---- uuuu
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
LVRC	0110 0110	0110 0110	uuuu uuuu
LVPUC	---- ---0	---- ---0	---- ---u
PSC0R	---- --00	---- --00	---- --uu
PSC1R	---- --00	---- --00	---- --uu
SCC	000- --00	000- --00	uuu- --uu
LVDC	0000 -000	0000 -000	uuuu -uuu
WDTC	0101 0011	0101 0011	uuuu uuuu
HIRCC	---- --01	---- --01	---- --uu
MFIO	--00 --00	--00 --00	--uu --uu
USR	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	uuuu uuuu
UCR3	---- ---0	---- ---0	---- ---u
BRDH	0000 0000	0000 0000	uuuu uuuu
BRDL	0000 0000	0000 0000	uuuu uuuu
UFCR	--00 0000	--00 0000	--uu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
RxCNT	---- -000	---- -000	---- -uuu
CRCCR	---- ---0	---- ---0	---- ---u
CRCIN	0000 0000	0000 0000	uuuu uuuu
CRCDL	0000 0000	0000 0000	uuuu uuuu
CRCDH	0000 0000	0000 0000	uuuu uuuu
CTM0C0	0000 0000	0000 0000	uuuu uuuu
CTM0C1	0000 0000	0000 0000	uuuu uuuu
CTM0DL	0000 0000	0000 0000	uuuu uuuu
CTM0DH	---- --00	---- --00	---- --uu
CTM0AL	0000 0000	0000 0000	uuuu uuuu
CTM0AH	---- --00	---- --00	---- --uu
CTM1C0	0000 0000	0000 0000	uuuu uuuu
CTM1C1	0000 0000	0000 0000	uuuu uuuu
CTM1DL	0000 0000	0000 0000	uuuu uuuu
CTM1DH	---- --00	---- --00	---- --uu
CTM1AL	0000 0000	0000 0000	uuuu uuuu
CTM1AH	---- --00	---- --00	---- --uu
INTEG	---- 0000	---- 0000	---- uuuu
EEAL	0000 0000	0000 0000	uuuu uuuu
EEAH	---- ---0	---- ---0	---- ---u
EED	0000 0000	0000 0000	uuuu uuuu
TKTMR	0000 0000	0000 0000	uuuu uuuu
TKC0	0000 0010	0000 0010	uuuu uuuu
TKC1	000- 0011	000- 0011	Uuu- uuuu

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TKC2	---- 0001	---- 0001	---- uuuu
TK16DL	0000 0000	0000 0000	uuuu uuuu
TK16DH	0000 0000	0000 0000	uuuu uuuu
TKM0C0	--0- 0000	--0- 0000	--u- uuuu
TKM0C1	0-00 0000	0-00 0000	u-uu uuuu
TKM0C2	1110 0100	1110 0100	uuuu uuuu
TKM016DL	0000 0000	0000 0000	uuuu uuuu
TKM016DH	0000 0000	0000 0000	uuuu uuuu
TKM0ROL	0000 0000	0000 0000	uuuu uuuu
TKM0ROH	---- --00	---- --00	---- --uu
TKM0TH16L	0000 0000	0000 0000	uuuu uuuu
TKM0TH16H	0000 0000	0000 0000	uuuu uuuu
TKM0THS	0000 0000	0000 0000	uuuu uuuu
PWRC	0000 0010	0000 0010	uuuu uuuu
PGAC0	0000 0000	0000 0000	uuuu uuuu
PGAC1	000- -000	000- -000	uuu- -uuu
PGACS	0000 0000	0000 0000	uuuu uuuu
ADRL	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRM	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR0	0000 0000	0000 0000	uuuu uuuu
ADCR1	--00 00--	--00 00--	--uu uu--
ADCS	00-0 0000	00-0 0000	uu-u uuuu
STKPTR	0--- 0000	0--- 0000	0--- 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	---0 0000	---0 0000	---u 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
IICC0	---- 000-	---- 000-	---- uuuu-
IICC1	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	0000 000-	0000 000-	uuuu uuuu-
IICTOC	0000 0000	0000 0000	uuuu uuuu
IECC	0000 0000	0000 0000	uuuu uuuu
PCRL	0000 0000	0000 0000	uuuu uuuu
PCRH	---0 0000	---0 0000	---u uuuu
ORMC	0000 0000	0000 0000	0000 0000
TMR	0000 0000	0000 0000	uuuu uuuu

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TMRC	--00 -000	--00 -000	--uu -uuu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	--00 0000	--00 0000	--uu uuuu
PBS0	0000 0000	0000 0000	uuuu uuuu
PMPS	--00 0000	--00 0000	--uu uuuu
TLVRC	---- --01	---- --01	---- --uu
IFS	---0 0000	---0 0000	---u uuuu
EEC	0000 0000	0000 0000	uuuu uuuu

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

Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	—	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	—	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	—	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	—	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	—	—	PB3	PB2	PB1	PB0
PBC	—	—	—	—	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	—	—	PBPU3	PBPU2	PBPU1	PBPU0
LVPUC	—	—	—	—	—	—	—	LVPUC

“—”: Unimplemented, read as “0”

I/O Logic Function Register List

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the PxPU and LVPUC registers, and are implemented using weak PMOS transistors. The PxPU register is used to determine whether the pull-high function is enabled or not while the LVPUC register is used to select the pull-high resistor value for low voltage power supply applications.

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

• **PxPU Register**

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

“—”: Unimplemented, read as “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 or B. However, the actual available bits for each I/O Port may be different.

• **LVPUC Register**

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

Bit 7~1 Unimplemented, read as “0”

Bit 0 **LVPU:** Pull-high resistor selection for low voltage power supply

0: All pin pull-high resistors are 60kΩ @ 3V

1: All pin pull-high resistors are 15kΩ @ 3V

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

Port A Wake-up

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

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

• **PAWU Register**

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

Bit 7 Unimplemented, read as “0”

Bit 6~0 **PAWU6~PAWU0:** PA6~PA0 wake-up function control

0: Disable

1: Enable

I/O Port Control Register

Each I/O Port has its own control register known as PAC~PBC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a “1”. This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a “0”, the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register.

However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin when the IECM is cleared to “0”.

• PxC Register

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

“—”: Unimplemented, read as “0”

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 or B. 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 PA0, PA2 and PA5. With the exception of RES/OCDS, the multi-power function is only effective when the pin is set to have a digital input or output function.

The port power can come from either the power pin VDD or VDDIO, which is determined using the PMPS5~PMPS0 bits in the PMPS register. The VDDIO power pin function should first be selected using the corresponding pin-shared function selection bits if the port power is supposed to come from the VDDIO pin.

An important point to know is that the input power voltage on the VDDIO pin should be equal to or less than the device supply power voltage V_{DD} when the VDDIO pin is selected as the port power supply pin. When the V_{DDIO} is less than 2.2V, it is recommended to be used when the V_{DD} is equal to the V_{DDIO} . Because when the V_{DDIO} is less than 2.2V, the I_{OH} thrust is too low when the V_{DD} is larger to the V_{DDIO} .

• PMPS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PMPS5	PMPS4	PMPS3	PMPS2	PMPS1	PMPS0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5~4 **PMPS5~PMPS4**: PA5 pin power supply selection

0x: V_{DD}

1x: V_{DDIO}

If the PA4 pin is switched to the VDDIO function, and the PMPS5~PMPS4 bits are set to “1x”, the VDDIO pin input voltage can be used for PA5 pin power.

- Bit 3~2 **PMPS3~PMPS2**: PA2 pin power supply selection
 0x: V_{DD}
 1x: V_{DDIO}
 If the PA4 pin is switched to the VDDIO function, and the PMPS3~PMPS2 bits are set to “1x”, the VDDIO pin input voltage can be used for PA2 pin power.
- Bit 1~0 **PMPS1~PMPS0**: PA0 pin power supply selection
 0x: V_{DD}
 1x: V_{DDIO}
 If the PA4 pin is switched to the VDDIO function, and the PMPS1~PMPS0 bits are set to “1x”, the VDDIO pin input voltage can be used for PA0 pin power.

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as INT_n, CTCK_n, 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	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	—	—	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
IFS	—	—	—	SDAPS1	SDAPS0	SCLPS	INT1PS	INT0PS

Pin-shared Function Selection Register List

• **PAS0 Register**

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

- Bit 7~6 **PAS07~PAS06:** PA3 pin-shared function selection
 00: PA3
 01: AIN5
 10: CTP1
 11: CTP0B
- Bit 5~4 **PAS05~PAS04:** PA2 pin-shared function selection
 00: PA2
 01: SDA
 10: RX/TX
 11: PA2
- Bit 3~2 **PAS03~PAS02:** PA1 pin-shared function selection
 00: PA1
 01: AIN4
 10: CTP0
 11: CTP1B
- Bit 1~0 **PAS01~PAS00:** PA0 pin-shared function selection
 00: PA0/INT0
 01: SCL
 10: TX
 11: PA0/INT0

• **PAS1 Register**

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

- Bit 7~6 Unimplemented, read as “0”
- Bit 5~4 **PAS15~PAS14:** PA6 pin-shared function selection
 00: PA6/INT0
 01: AIN6
 10: CTP0
 11: SDA
- Bit 3~2 **PAS13~PAS12:** PA5 pin-shared function selection
 00: PA5/INT1/CTCK0
 01: RCFIN
 10: CTP1
 11: SCL
- Bit 1~0 **PAS11~PAS10:** PA4 pin-shared function selection
 00: PA4/INT1/CTCK1
 01: VDDIO
 10: SDA
 11: PA4/INT1/CTCK1

• **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: KEY4
- Bit 5~4 **PBS05~PBS04:** PB2 pin-shared function selection
00: PB2
01: PB2
10: PB2
11: KEY3
- Bit 3~2 **PBS03~PBS02:** PB1 pin-shared function selection
00: PB1
01: PB1
10: PB1
11: KEY2
- Bit 1~0 **PBS01~PBS00:** PB0 pin-shared function selection
00: PB0
01: RCFIN
10: PB0
11: KEY1

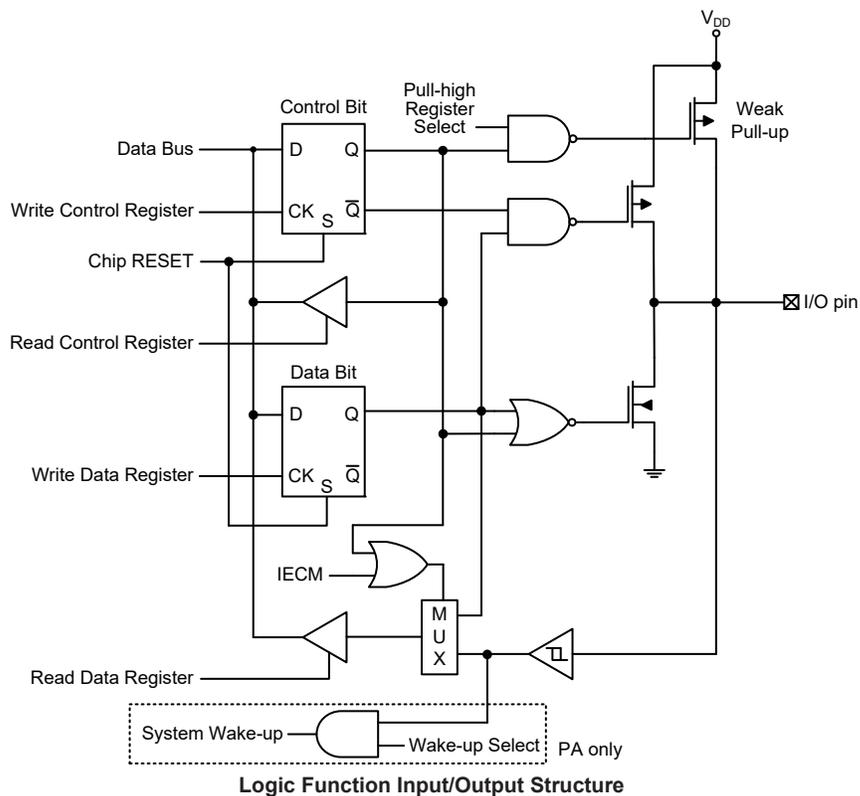
• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	SDAPS1	SDAPS0	SCLPS	INT1PS	INT0PS
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

- Bit 7~5 Unimplemented, read as “0”
- Bit 4~3 **SDAPS1~SDAPS0:** SDA input source pin selection
00: PA2
01: PA4
10: PA6
11: PA2
- Bit 2 **SCLPS:** SCL input source pin selection
0: PA0
1: PA5
- Bit 1 **INT1PS:** INT1 input source pin selection
0: PA4
1: PA5
- Bit 0 **INT0PS:** INT0 input source pin selection
0: PA0
1: PA6

I/O Pin Structures

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



READ PORT Function

The READ PORT function is used to read data from I/O pins, which is specially designed for the IEC 60730 self-diagnostic test on the I/O function and A/D paths. After the self-diagnostic test on the I/O function and A/D paths are completed, users must disable the READ PORT function immediately. In cases other than the I/O function and A/D path self-diagnostic test, it is strongly recommended to disable the READ PORT function to avoid affecting other peripheral functions and causing unexpected consequences.

There is a register, IECC, which is used to control the READ PORT function. When a specific data pattern, "11001010", is written into the IECC register, the internal signal named IECM will be set high to enable the READ PORT function. If the READ PORT function is enabled, the reading path is from the I/O pins. The value on the corresponding pins will be passed to the accumulator ACC when the read port instruction "mov a, Px" is executed, where the "x" stands for the corresponding I/O port name. However, when the IECC register content is set to any other values rather than "11001010", the IECM internal signal will be cleared to 0 to disable the READ PORT function, and the reading path will be from the data latch or I/O pins. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function.

• **IECC Register**

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

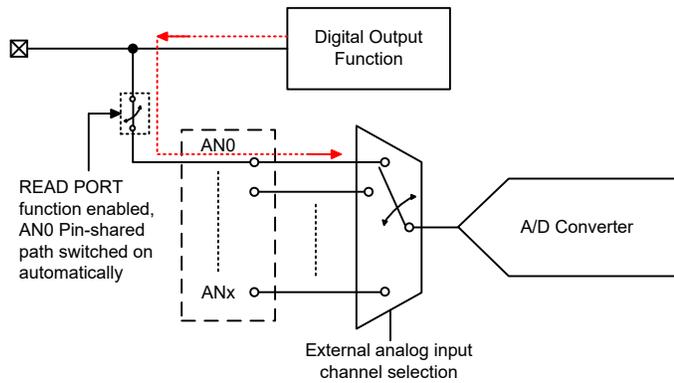
Bit 7~0 **IECS7~IECS0**: READ PORT function enable control bit 7~ bit 0
 11001010: IECSM=1 – READ PORT function is enabled
 Others: IECSM=0 – READ PORT function is disabled

READ PORT Function Port Control Register Bit – PxC.n	Disabled		Enabled	
	1	0	1	0
I/O Function	Pin value	Data latch value	Pin value	
A/D Function	0			

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

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

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



A/D Channel Input Path Internal Connection

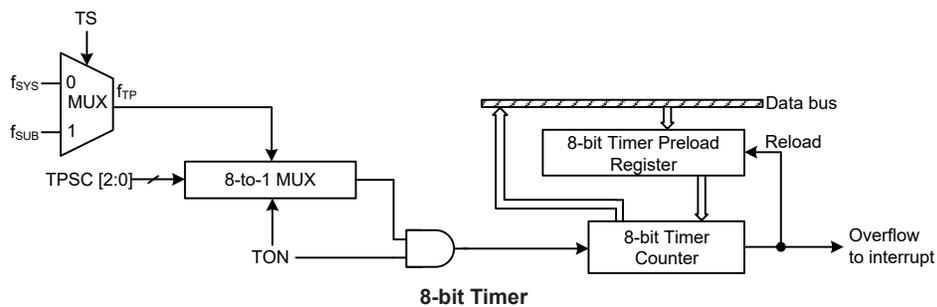
Programming Considerations

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

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

Timer

The provision of the Timer forms an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contains an 8-bit Timer, which contains an 8-bit programmable count-up counter and the clock comes from an internal clock source.



Timer Clock Source

The clock source can be configured by the TS bit in the TMRC Timer Control Register to be derived from the f_{SYS} or f_{SUB} clock, the division ratio of which is selected by the TPSC2~TPSC0 bits in the TMRC Timer Control Register.

Timer Registers

There are two registers related to the Timer. The first is the TMR register that contains the actual value of the timer and into which an initial value can be preloaded. Writing to the TMR register will transfer the specified data to the Timer. Reading the TMR register will read the contents of the Timer. The second is the TMRC control register, which is used to select the internal clock source, control the timer enable or disable and select the timer prescaler rate.

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMR	D7	D6	D5	D4	D3	D2	D1	D0
TMRC	—	—	TS	TON	—	TPSC2	TPSC1	TPSC0

Timer Register List

Timer Register – TMR

The timer register TMR is the place where the actual timer value is stored. The value in the timer register increases by one each time an internal clock pulse is received. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit Timer, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be loaded with the preload register value and continue counting.

Note that to achieve a maximum full range count of FFH, the preload register must first be cleared. Note that if the Timer is in an off condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter until an overflow occurs.

• TMR Register

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

Bit 7~0 **D7~D0**: Timer preload register byte

Timer Control Register – TMRC

The Timer Control Register is known as TMRC. It is the Timer Control Register together with its corresponding timer register that controls the full operation of the Timer. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

The timer-on bit TON provides the basic on/off control of the respective timer. Setting the bit to high allows the counter to run. Clearing the bit stops the counter. When the internal clock source is used, it can be sourced from the f_{SYS} or f_{SUB} clock selected by setting the TS bit. Bits TPSC2~TPSC0 determine the division ratio of the selected clock source.

• TMRC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TS	TON	—	TPSC2	TPSC1	TPSC0
R/W	—	—	R/W	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5 **TS**: Timer clock source
 0: f_{SYS}
 1: f_{SUB}

Bit 4 **TON**: Timer enable control
 0: Disable
 1: Enable

Bit 3 Unimplemented, read as “0”

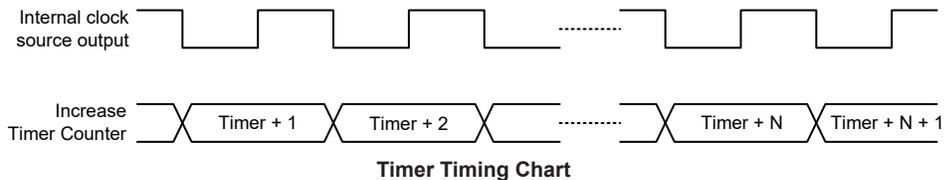
Bit 2~0 **TPSC2~TPSC0**: Timer prescaler rate f_{Timer} selection
 000: $f_{TP}/2^0$
 001: $f_{TP}/2^1$
 010: $f_{TP}/2^2$
 011: $f_{TP}/2^3$
 100: $f_{TP}/2^4$
 101: $f_{TP}/2^5$

110: $f_{TP}/2^6$
 111: $f_{TP}/2^7$
 $t_{Timer} = 1/f_{Timer}$.

Timer Operation

The Timer can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer overflows.

The internal clock f_{TP} is used as the timer clock, which can be selected to be derived from f_{SYS} or f_{SUB} by setting the TS bit in the TMRC register. The division of the f_{TP} clock is selected by the TPSC2~TPSC0 bits in the same register. The timer-on bit TON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increases by one. When the timer reaches its maximum 8-bit, FFH Hex, value and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. It should be noted that in the Timer mode, even if the device is in the IDLE/SLEEP mode, if the selected internal clock is still activated and a timer overflow occurs, it will generate a timer interrupt and corresponding wake-up source.



Programming Considerations

As the internal system clock is used as the timer clock source, the timer can be synchronised with the overall operation of the microcontroller. When the timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector.

When the Timer is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, it should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialised before using them for the first time. The associated timer interrupt enable bit in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The clock source control, timer enable control and timer prescaler rate selection bits in timer control register must also be correctly issued to ensure the timer is properly configured for the required applications. It is also important to ensure that a desired initial value is first loaded into the timer register before the timer is switched on. After the timer has been initialised the timer can be turned on and off by controlling the enable bit in the timer control register.

When the Timer overflows, its corresponding interrupt request flag in the interrupt control register will be set to generate an interrupt signal. If the Timer interrupt is enabled this will in turn allow program branch to its interrupt vector. However irrespective of whether the interrupt is enabled or not, a Timer overflow will also generate a wake-up signal if the device is in the IDLE/SLEEP mode. This situation may occur if the Timer internal clock source is still activated. In such cases, the Timer will continue to count and if an overflow occurs the device will be woken up. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the "HALT" instruction to enter the IDLE/SLEEP mode.

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 as well as being the functional unit for the generation of PWM signals. Each of the TMs has two individual interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

The general features of the Compact Type TM are described here with more detailed information provided in the individual Compact Type TM section.

Introduction

The device contains two Compact Type TMs. The main features of the CTM are summarised in the accompanying table.

TM Function	CTM
Timer/Counter	√
Compare Match Output	√
PWM Output	√
PWM Alignment	Edge
PWM Adjustment Period & Duty	Duty or Period

TM Function Summary

TM Operation

The Compact Type TMs offer a diverse range of functions, from simple timing operations to PWM signal generation. The key to understanding how the TM operates is to see it in terms of a free running counter whose value is then compared with the value of pre-programmed internal comparators. When the free running 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 CTnCK2~CTnCK0 bits in the CTMn control registers, where “n” stands for the specific TM serial number. The clock source can be a ratio of the system clock f_{SYS} or the internal high clock f_H , the f_{SUB} clock source or the external CTCKn pin. The CTCKn pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

Each Compact Type TM has two internal interrupts, the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs. When a TM interrupt is generated, it can be used to clear the counter and also to change the state of the TM output pin.

TM External Pins

Each of the Compact Type TMs has one TM input pin, with the label CTCKn. The CTMn input pin, CTCKn, is essentially a clock source for the CTMn and is selected using the CTnCK2~CTnCK0

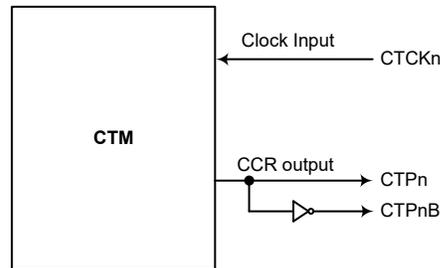
bits in the CTMnCO register. This external TM input pin allows an external clock source to drive the internal TM. The CTCKn input pin can be chosen to have either a rising or falling active edge.

Each CTM has two output pins with the label CTPn and CTPnB. When the TM is in the Compare Match Output Mode, the CTPn pin can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The CTPnB pin outputs the inverted signal of the CTPn. The external CTPn and CTPnB output pins are also the pins where the TM generates the PWM output waveform.

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

CTM0		CTM1	
Input	Output	Input	Output
CTCK0	CTP0, CTP0B	CTCK1	CTP1, CTP1B

TM External Pins

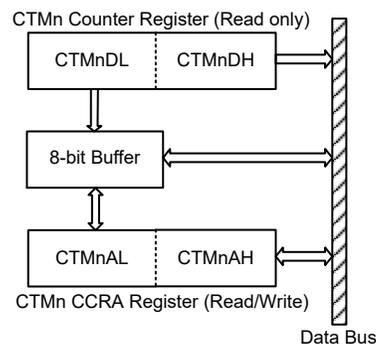


CTM Function Pin Block Diagram (n=0~1)

Programming Considerations

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

As the CCRA registers are implemented in the way shown in the following diagram and accessing this register pair is carried out in a specific way as described above, it is recommended to use the “MOV” instruction to access the CCRA low byte register, named CTMnAL, in the following access procedures. Accessing the CCRA low byte register without following these access procedures will result in unpredictable values.

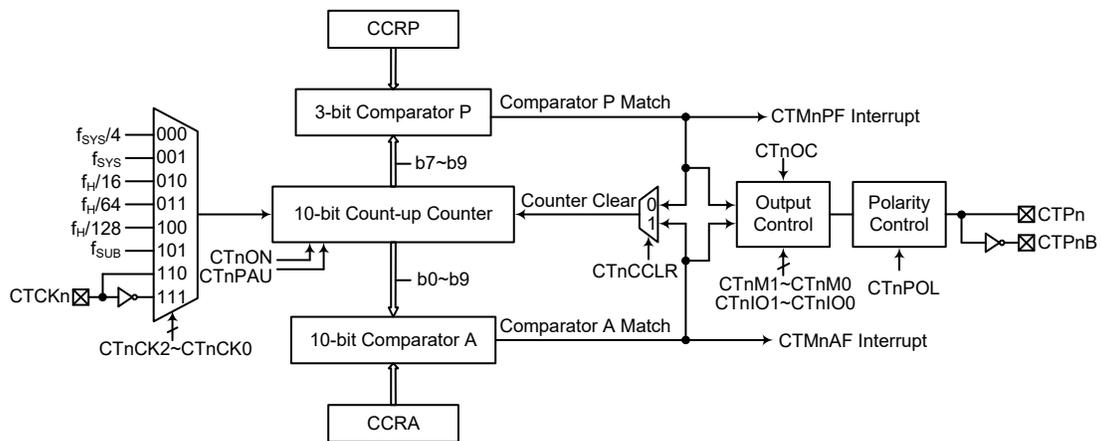


The following steps show the read and write procedures:

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

Compact Type TM – CTM

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



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

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

Compact Type TM Operation

The size of Compact TM is 10-bit wide and its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP comparator is 3-bit wide whose value is compared with the highest 3 bits in the counter while the CCRA is the 10 bits and therefore compares all counter bits. The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the CTnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators.

When these conditions occur, a CTMn interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

Compact Type TM Register Description

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

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

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

• CTMnC0 Register

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

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

Bit 6~4 **CTnCK2~CTnCK0:** CTMn counter clock selection
 000: $f_{SYS}/4$
 001: f_{SYS}
 010: $f_H/16$
 011: $f_H/64$
 100: $f_H/128$
 101: f_{SUB}
 110: CTCKn rising edge clock
 111: CTCKn falling edge clock
 These three bits are used to select the clock source for the CTMn. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the Operating Modes and System Clocks section.

Bit 3 **CTnON:** CTMn counter on/off control
 0: Off
 1: On
 This bit controls the overall on/off function of the CTMn. Setting the bit high enables the counter to run while clearing the bit disables the CTMn. Clearing this bit to zero

will stop the counter from counting and turn off the CTMn which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.

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

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

Comparator P match period=
 000: 1024 CTMn clocks
 001~111: (1~7)×128 CTMn clocks

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

• **CTMnC1 Register**

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

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

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

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

Compare Match Output Mode
 00: No change
 01: Output low
 10: Output high
 11: Toggle output
 PWM Output Mode
 00: PWM output inactive state
 01: PWM output active state
 10: PWM output
 11: Undefined
 Timer/Counter Mode
 Unused

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

In the Compare Match Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a compare match occurs from the Comparator A.

The CTMn output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the CTMn output pin should be setup using the CTnOC bit in the CTMnC1 register. Note that the output level requested by the CTnIO1 and CTnIO0 bits must be different from the initial value setup using the CTnOC bit otherwise no change will occur on the CTMn output pin when a compare match occurs. After the CTMn output pin changes state, it can be reset to its initial level by changing the level of the CTnON bit from low to high.

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

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

Compare Match Output Mode

0: Initial low

1: Initial high

PWM Output Mode/Single Pulse Output Mode

0: Active low

1: Active high

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

Bit 2 **CTnPOL:** CTMn CTPn output polarity control

0: Non-inverted

1: Inverted

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

Bit 1 **CTnDPX:** CTMn PWM 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 **CTCCLR:** CTMn 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 CTMn contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the CTnCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The CTnCCLR bit is not used in the PWM Output mode.

• **CTMnDL Register**

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

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

• **CTMnDH Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **D9~D8**: CTMn Counter High Byte Register bit 1 ~ bit 0

• **CTMnAL Register**

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

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

• **CTMnAH Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 7~0 **D9~D8**: CTMn CCRA High Byte Register bit 1 ~ bit 0

Compact Type TM Operation Modes

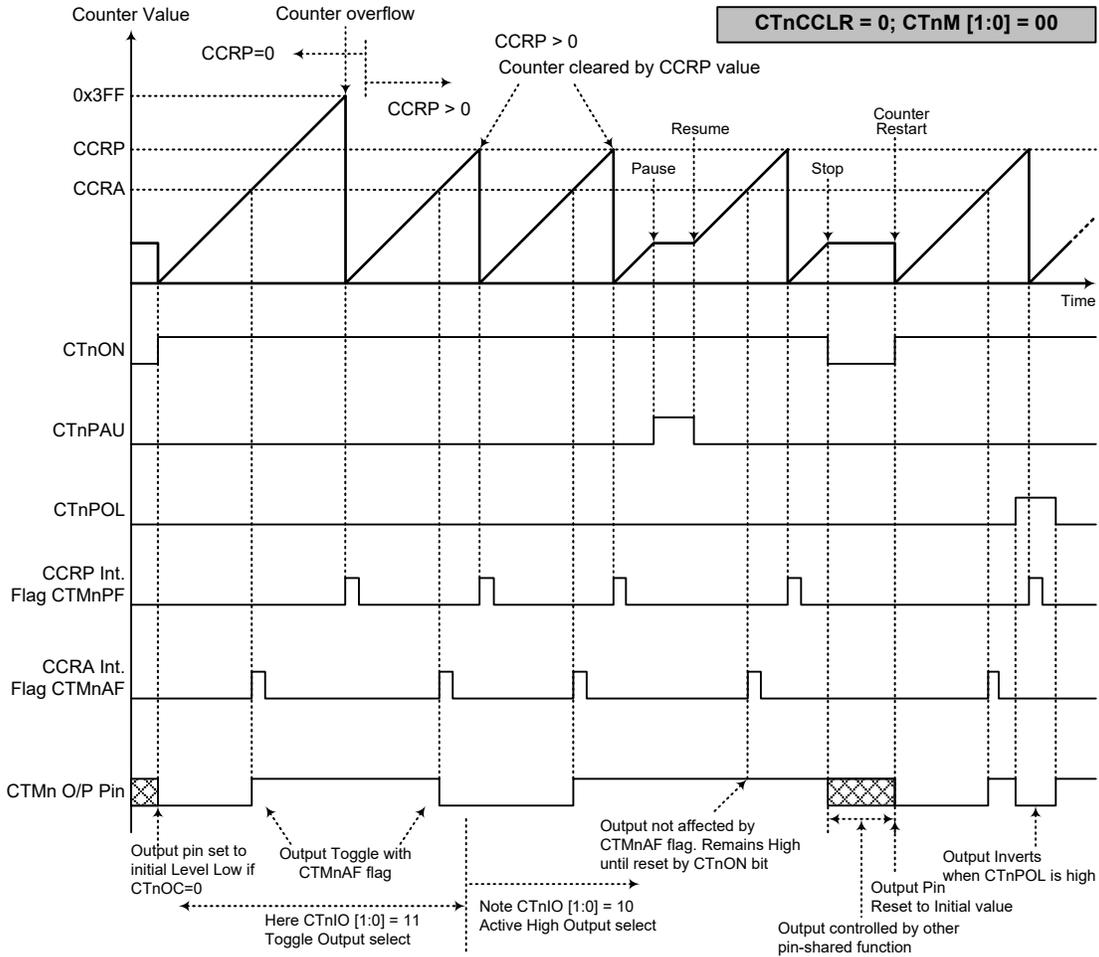
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

Compare Match Output Mode

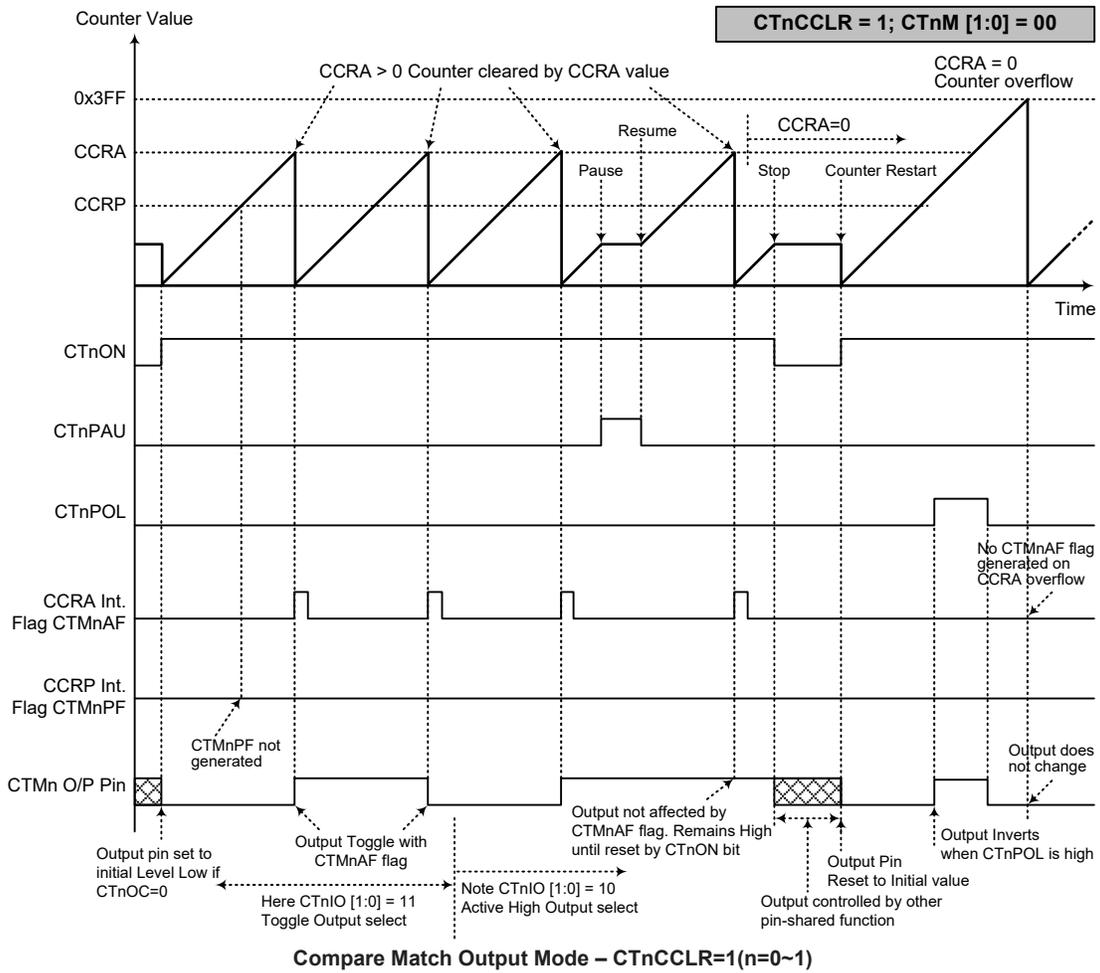
To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMnAF and CTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

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

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



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



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

Timer/Counter Mode

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

PWM Output Mode

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

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the CTnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTnDPX bit in the CTMnC1 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 CTnOC bit in the CTMnC1 register is used to select the required polarity of the PWM waveform while the two CTnIO1 and CTnIO0 bits are used to enable the PWM output or to force the CTMn output pin to a fixed high or low level. The CTnPOL bit is used to reverse the polarity of the PWM output waveform.

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

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

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

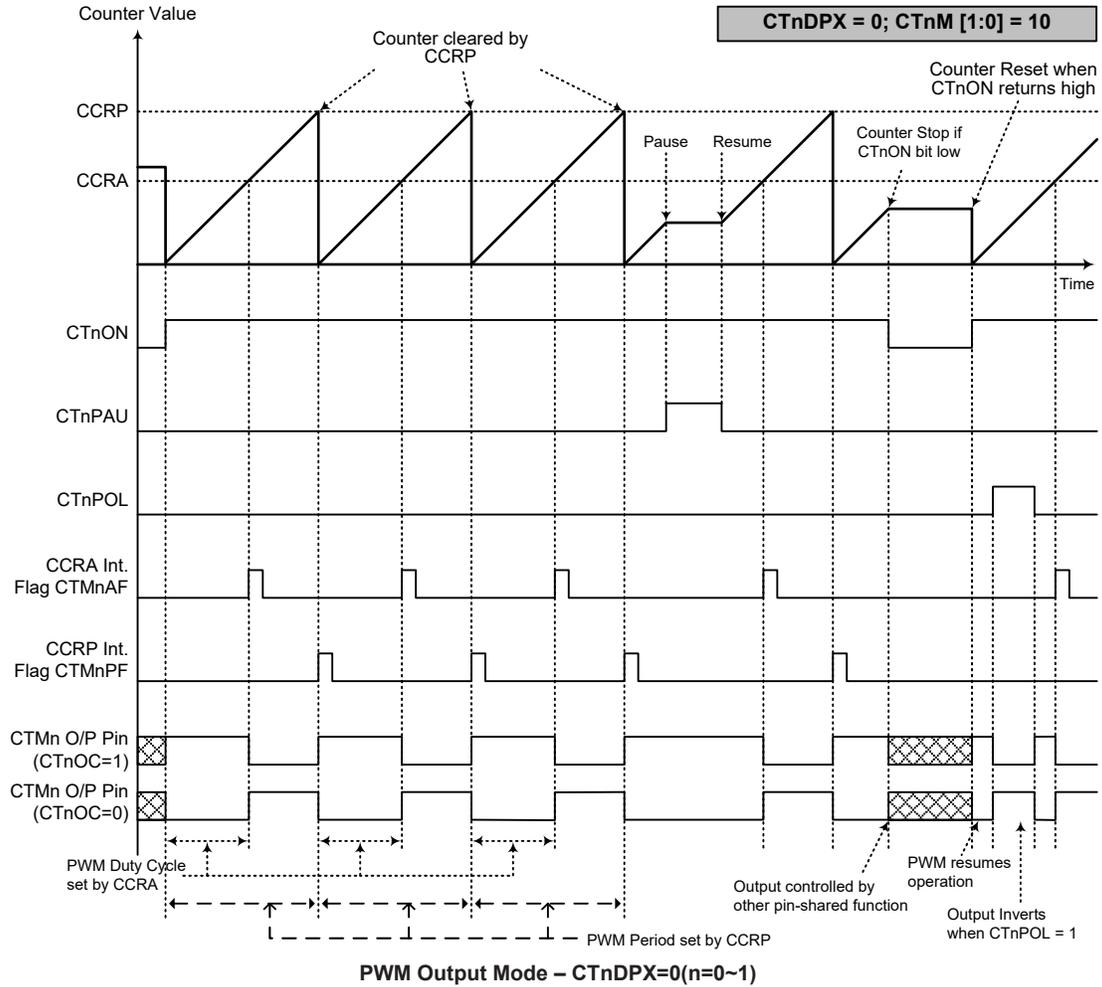
The CTMn PWM output frequency= $(f_{SYS}/4)/(4\times 128)=f_{SYS}/2048=4\text{kHz}$, duty= $128/(4\times 128)=25\%$.

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

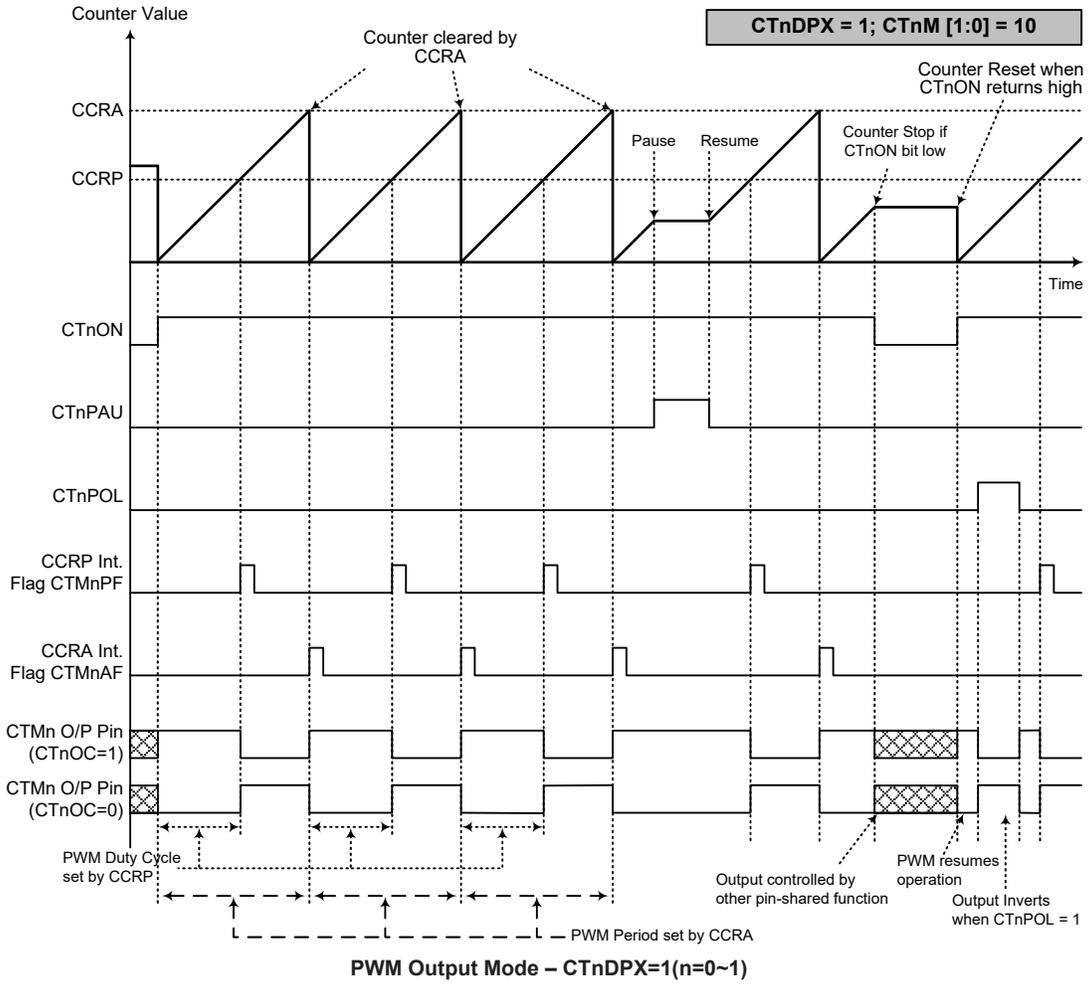
• **10-bit CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=1**

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

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



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



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

Touch Key Function

The device provides multiple touch key functions. The touch key function is fully integrated and requires no external components, allowing touch key functions to be implemented by the simple manipulation of internal registers.

Touch Key Structure

The touch keys are pin shared with the I/O pins, with the desired function chosen via the pin-shared selection register bit. Keys are organised into one group, known as a module. The module is a fully independent set of four Touch Keys and has its own oscillator. The module contains its own control logic circuits and register set.

Total Key Number	Touch Key	Shared I/O Pin
4	KEY1~KEY4	PB0~PB3

Touch Key Structure

Touch Key Register Definition

The touch key module 0, which contains four touch key functions, is controlled using several registers. The following table shows the register set for the touch key module 0.

Register Name	Description
TKTMR	Touch key time slot 8-bit counter preload register
TKC0	Touch key function control register 0
TKC1	Touch key function control register 1
TKC2	Touch key function control register 2
TK16DL	Touch key function 16-bit counter low byte
TK16DH	Touch key function 16-bit counter high byte
TKM0C0	Touch key module 0 control register 0
TKM0C1	Touch key module 0 control register 1
TKM0C2	Touch key module 0 control register 2
TKM016DL	Touch key module 0 16-bit C/F counter low byte
TKM016DH	Touch key module 0 16-bit C/F counter high byte
TKM0ROL	Touch key module 0 reference oscillator capacitor selection low byte
TKM0ROH	Touch key module 0 reference oscillator capacitor selection high byte
TKM0TH16L	Touch key module 0 16-bit threshold low byte
TKM0TH16H	Touch key module 0 16-bit threshold high byte
TKM0THS	Touch key module 0 threshold comparison flag

Touch Key Function Register Definition

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	TKRAMC	TKRCOV	TKST	TKCFOV	TK16OV	TKMOD1	TKMOD0	TKBUSY
TKC1	D7	D6	D5	—	TK16S1	TK16S0	TKFS1	TKFS0
TKC2	—	—	—	—	TSSS	TSC	ASMP1	ASMP0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKM0C0	—	—	MODFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
TKM0C1	M0TSS	—	M0ROEN	M0KOEN	M0K4EN	M0K3EN	M0K2EN	M0K1EN
TKM0C2	M0SK31	M0SK30	M0SK21	M0SK20	M0SK11	M0SK10	M0SK01	M0SK00

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKM016DL	D7	D6	D5	D4	D3	D2	D1	D0
TKM016DH	D15	D14	D13	D12	D11	D10	D9	D8
TKM0ROL	D7	D6	D5	D4	D3	D2	D1	D0
TKM0ROH	—	—	—	—	—	—	D9	D8
TKM0TH16L	D7	D6	D5	D4	D3	D2	D1	D0
TKM0TH16H	D15	D14	D13	D12	D11	D10	D9	D8
TKM0THS	M0K4THF	M0K3THF	M0K2THF	M0K1THF	M0K4THS	M0K3THS	M0K2THS	M0K1THS

Touch Key Function Register List

• **TKTMR Register**

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

Bit 7~0 **D7~D0:** Touch key time slot 8-bit counter preload register

The touch key time slot counter preload register is used to determine the touch key time slot overflow time. The time slot unit period is equal to t_{STC} or $32t_{STC}$. It can be obtained by a 5-bit counter and equal to 32 time slot clock cycles. It also can be obtained by bypassing a 5-bit counter and equal to 1 time slot clock cycle. Therefore, the time slot counter overflow time is equal to the following equation shown.

If $M0TSS=0$, Time slot counter overflow time= $(256-TKTMR[7:0]) \times 32t_{TSC}$, while if $M0TSS=1$, Time slot counter overflow time= $(256-TKTMR[7:0]) \times t_{TSC}$, where t_{TSC} is the time slot counter clock period.

• **TKC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	TKRAMC	TKRCOV	TKST	TKCFOV	TK16OV	TKMOD1	TKMOD0	TKBUSY
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
POR	0	0	0	0	0	0	1	0

Bit 7 **TKRAMC:** Touch key data memory access control

- 0: Accessed by MCU
- 1: Accessed by touch key module

This bit determines that the touch key data memory is used by the MCU or the touch key module. However, the touch key module will have the priority to access the touch key data memory when the touch key module operates in the auto scan mode or the periodic auto scan mode, i.e., the TKST bit state is changed from 0 to 1 when the TKMOD1~TKMOD0 bits are set to 00, 10 or 11. After the touch key auto scan or the periodic auto scan operation is completed, i.e., the TKBUSY bit state is changed from 1 to 0, the touch key data memory access will be controlled by the TKRAMC bit. Therefore, it is recommended to set the TKRAMC bit to 1 when the touch key module operates in the auto scan mode or the periodic auto scan mode. Otherwise, the contents of the touch key data memory may be modified as this data memory space is configured by the touch key module followed by the MCU access.

Bit 6 **TKRCOV:** Touch key time slot counter overflow flag

- 0: No overflow occurs
- 1: Overflow occurs

This bit can be accessed by application program. When this bit is set by touch key time slot counter overflow, the corresponding touch key TKRCOV interrupt request flag will be set. However, if this bit is set by application program, the touch key

TKRCOV interrupt request flag will not be affected. Therefore, this bit can not be set by application program but must be cleared to 0 by application program.

In the auto scan mode, if the time slot counter overflows but the touch key auto scan operation is not completed, the TKRCOV bit will not be set. At this time, the touch key module 16-bit C/F counter, touch key function 16-bit counter and 5-bit time slot unit period counter will be automatically cleared but the 8-bit time slot counter will be reloaded from the 8-bit time slot counter preload register. When the touch key auto scan operation is completed, the TKRCOV bit and the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set and all module keys and reference oscillators will automatically stop. The touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off.

In the periodic auto scan mode, the TKRCOV bit is cleared to zero during the auto scan operation period. Only at the end of the last scan operation in the WDT time-out cycle, the 16-bit C/F counter content will be written into the corresponding touch key data memory, and then the TKRCOV bit will be set high by the hardware circuit. The other actions in this mode are the same as those in the auto scan mode except the above mentioned.

In the manual scan mode, if the time slot counter overflows, the TKRCOV bit and the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set and all module keys and reference oscillators will automatically stop. The touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off.

Bit 5 **TKST**: Touch key detection start control
 0: Stopped or no operation
 0→1: Start detection

The touch key module 0 16-bit C/F counter, touch key function 16-bit counter and 5-bit time slot unit period counter will automatically be cleared when this bit is cleared to zero. However, the 8-bit programmable time slot counter will not be cleared. When this bit is changed from low to high, the touch key module 0 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be switched on together with the key and reference oscillators to drive the corresponding counters.

Bit 4 **TKCFOV**: Touch key module 16-bit C/F counter overflow flag
 0: No overflow occurs
 1: Overflow occurs

This bit is set high by the touch key module 16-bit C/F counter overflow and must be cleared to 0 by application programs.

Bit 3 **TK16OV**: Touch key function 16-bit counter overflow flag
 0: No overflow occurs
 1: Overflow occurs

This bit is set high by the touch key function 16-bit counter overflow and must be cleared to 0 by application programs.

Bit 2~1 **TKMOD1~TKMOD0**: Touch key scan mode selection
 00: Auto scan mode
 01: Manual scan mode
 10/11: Periodic auto scan mode

In the manual scan mode, the reference oscillator capacitor value should be properly configured before the scan operation begins and the touch key module 0 16-bit C/F counter value should be read after the scan operation finishes by application program. In the auto scan mode, the data movement which is described above is implemented by hardware. The individual reference oscillator capacitor value and 16-bit C/F counter content for all scanned keys will be read from a dedicated Touch Key Data Memory area. In the auto scan mode, the keys to be scanned can be arranged in a specific sequence which is determined by the M0SK3[1:0]~M0SK0[1:0] bits in the TKM0C2 register. The scan operation will not be stopped until all arranged keys are scanned.

In the periodic auto scan mode, the touch key scan operation will be implemented automatically on a periodic basis, which can be determined by the ASMP1~ASMP0 bits in the TKC2 register. Only at the end of the last scan operation in the WDT timeout cycle, the 16-bit C/F counter content for all scanned keys will be written into the corresponding touch key data memory. In addition, when any key C/F counter value is less than the lower threshold if M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1, the TKTH signal will be set high. The other actions in this mode are the same as those in the auto scan mode except the above mentioned.

- Bit 0** **TKBUSY**: Touch key scan operation busy flag
 0: Not busy – no scan operation is executed or scan operation is completed
 1: Busy – scan operation is executing

This bit indicates whether the touch key scan operation is executing or not. It is set to 1 when the TKST bit is set high to start the scan operation.

In the manual scan mode this bit is cleared to 0 automatically when the touch key time slot counter overflows. In the auto scan mode this bit is cleared to 0 automatically when the touch key scan operation is completed. In the periodic auto scan mode this bit is cleared to 0 automatically when the last scan operation in the WDT time-out cycle is completed, or when any key C/F counter value is less than the lower threshold if M0KnTHS=0, or when the value is larger than the upper threshold if M0KnTHS=1.

• **TKC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	—	TK16S1	TK16S0	TKFS1	TKFS0
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	0	0	0	—	0	0	1	1

- Bit 7~5** **D7~D5**: Data bits for test only
 These bits are used for test purpose only and must be kept as “000” for normal operations.

- Bit 4** Unimplemented, read as “0”

- Bit 3~2** **TK16S1~TK16S0**: Touch key function 16-bit counter clock source selection
 00: f_{SYS}
 01: $f_{SYS}/2$
 10: $f_{SYS}/4$
 11: $f_{SYS}/8$

- Bit 1~0** **TKFS1~TKFS0**: Touch key oscillator and Reference oscillator frequency select
 00: 1MHz
 01: 3MHz
 10: 7MHz
 11: 11MHz

• **TKC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	TSSS	TSC	ASMP1	ASMP0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	1	1

- Bit 7~4** Unimplemented, read as “0”

- Bit 3** **TSSS**: Auto scan mode periode selection
 0: t_{WDT}
 1: $t_{TIMER} (2^5 \times t_{TIMER})$

Note: When TSSS=1, all the WDT time-out (t_{WDT}) mentioned in this chapter other than this register should be replaced by t_{TIMER} .

Bit 2 **TSC**: Time slot control
 0: Time slot 0~3
 1: Time slot 0 only

This bit is used to configure time slot, if TSC is set to “1”, only time slot 0 is active to execute the touch key related operations and time slot 1~3 are invalid. It can reduce power consumption and implement the 1-key wake-up function from IDLE or SLEEP mode. The desired wake-up key in time slot 0 is selected by the M0SK01~M0SK00 bits in the TKM0C2 register.

Bit 1~0 **ASMP1~ASMP0**: Periodic auto scan mode period selection

TSSS=0
 00: $2^{14}/f_{LIRC}$
 01: $2^{13}/f_{LIRC}$
 10: $2^{12}/f_{LIRC}$
 11: $2^{11}/f_{LIRC}$

These bits are used to determine the touch key scan period and only available when the touch key function is configured to operate in the periodic auto scan mode. The number of touch key scan times is obtained by the WDT time-out period, t_{WDT} , and the periodic auto scan mode period, t_{KEY} , using the equation, $N=t_{WDT}/t_{KEY}$. The WDT time-out period t_{WDT} is selected by the WS2~WS0 bits.

For example, if the WDT time-out period is $2^{15}/f_{LIRC}$ by setting the WS[2:0] bits to 100, then t_{WDT} is about equal to 1.024s. Therefore, the number of touch key scan times is 2/4/8/16 times in a WDT time-out cycle when the ASMP[1:0] bits are set to 00/01/10/11 respectively. It is extremely important to ensure that the periodic auto scan mode period t_{KEY} does not exceed the WDT time-out period t_{WDT} in applications.

TSSS=1
 00: $t_{Timer} \times 2^4$
 01: $t_{Timer} \times 2^3$
 10: $t_{Timer} \times 2^2$
 11: $t_{Timer} \times 2^1$

These bits are used to determine the touch key scan period and only available when the touch key function is configured to operate in the periodic auto scan mode.

For example, if the t_{Timer} is 50ms, then t_{TIMER} is about equal to $50ms \times 2^5$. Therefore, the number of touch key scan times is 2/4/8/16 times in a t_{TIMER} cycle when the ASMP[1:0] bits are set to 00/01/10/11 respectively. It is recommended that the t_{Timer} should be larger than 0.5ms in applications.

• **TK16DH/TK16DL – Touch Key Function 16-bit Counter Register Pair**

Register	TK16DH								TK16DL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register pair is used to store the touch key function 16-bit counter value. This 16-bit counter can be used to calibrate the reference or key oscillator frequency. When the touch key time slot counter overflows in the manual scan mode, this 16-bit counter will be stopped and the counter content will be unchanged. However, this 16-bit counter content will be cleared to zero at the end of the time slot 0, slot 1 and slot 2 but kept unchanged at the end of the time slot 3 in the auto scan mode or the periodic auto scan mode. This register pair will be cleared to zero when the TKST bit is set low.

• **TKM016DH/TKM016DL – Touch Key Module 0 16-bit C/F Counter Register Pair**

Register	TKM016DH								TKM016DL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register pair is used to store the touch key module 0 16-bit C/F counter value. This 16-bit C/F counter will be stopped and the counter content will be kept unchanged when the touch key time slot counter overflows in the manual scan mode. However, this 16-bit C/F counter content will be cleared to zero at the end of the time slot 0, slot 1 and slot 2 after it is written to the touch key data memory but kept unchanged at the end of the time slot 3 when the auto scan mode or the periodic auto scan mode is selected. This register pair will be cleared to zero when the TKST bit is cleared to zero.

• **TKM0ROH/TKM0ROL – Touch Key Module 0 Reference Oscillator Capacitor Selection Register Pair**

Register	TKM0ROH								TKM0ROL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	—	—	—	—	—	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	—	—	—	0	0	0	0	0	0	0	0	0	0

This register pair is used to store the touch key module 0 reference oscillator capacitor value. This register pair will be loaded with the corresponding next time slot capacitor value from the dedicated touch key data memory at the end of the current time slot when the auto scan mode or the periodic auto scan mode is selected.

The reference oscillator internal capacitor value=(TKM0RO[9:0]×50pF)/1024

• **TKM0C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	MODFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
R/W	—	—	R/W	—	R/W	R/W	R/W	R/W
POR	—	—	0	—	0	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5 **MODFEN**: Touch key module 0 multi-frequency control
0: Disable
1: Enable

This bit is used to control the touch key oscillator frequency doubling function. When this bit is set to 1, the key oscillator frequency will be doubled.

Bit 4 Unimplemented, read as “0”

Bit 3 **M0SOFC**: Touch key module 0 C-to-F oscillator frequency hopping function control selection
0: Controlled by the M0SOF2~M0SOF0
1: Controlled by hardware circuit

This bit is used to select the touch key oscillator frequency hopping function control method. When this bit is set to 1, the key oscillator frequency hopping function is controlled by the hardware circuit regardless of the M0SOF2~M0SOF0 bits value.

Bit 2~0 **M0SOF2~M0SOF0**: Touch key module 0 Reference and Key oscillators hopping frequency selection (when M0SOFC=0)
000: 1.020MHz
001: 1.040MHz

010: 1.059MHz
 011: 1.074MHz
 100: 1.085MHz
 101: 1.099MHz
 110: 1.111MHz
 111: 1.125MHz

These bits are used to select the touch key oscillator frequency for the hopping function. Note that these bits are only available when the M0SOFC bit is cleared to 0.

The frequencies mentioned here are only for the condition where the key and reference oscillator frequency is selected to be 1MHz, these values will be changed when the external or internal capacitor has different values. Users can adjust the key and reference oscillator frequency in scale when any other frequency is selected.

• **TKM0C1 Register**

Bit	7	6	5	4	3	2	1	0
Name	M0TSS	—	M0ROEN	M0KOEN	M0K4EN	M0K3EN	M0K2EN	M0K1EN
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 **M0TSS**: Touch key module 0 time slot counter clock source Selection

0: Touch key module 0 reference oscillator
 1: f_{LIRC}

When the M0TSS bit is high, the touch key module 0 time slot counter clock comes from the f_{LIRC} oscillator, which can remain on even when the device enters the IDLE and SLEEP modes. The touch key module 0 time slot counter clock is then can bypass the 5-bit counter to reduce the overflow time. In the periodic auto scan mode, set the M0TSS and TSC bits high can reduce the touch key standby power.

Note: If the 5-bit counter is bypassed, the touch key module 0 Reference and Key oscillators hopping frequency is selected by the M0SOF2~M0SOF0 bits regardless of the M0SOFC bit setting.

Bit 6 Unimplemented, read as “0”

Bit 5 **M0ROEN**: Touch key module 0 Reference oscillator enable control

0: Disable
 1: Enable

This bit is used to enable the touch key module 0 reference oscillator. In the auto scan mode or the periodic auto scan mode, the reference oscillator will automatically be enabled by setting the M0ROEN bit high when the TKST bit is set from low to high if the reference oscillator is selected as the time slot clock source. The combination of the M0TSS and M0K4EN~M0K1EN bits determines whether the reference oscillator is used or not. When the TKBUSY bit is changed from high to low, the M0ROEN bit will automatically be cleared to zero to disable the reference oscillator.

In the manual scan mode, the reference oscillator should first be enabled before setting the TKST bit from low to high if the reference oscillator is selected to be used and will be disabled when the TKBUSY bit is changed from high to low.

Bit 4 **M0KOEN**: Touch key module 0 Key oscillator enable control

0: Disable
 1: Enable

This bit is used to enable the touch key module 0 key oscillator. In the auto scan mode or the periodic auto scan mode, the key oscillator will automatically be enabled by setting the M0KOEN bit high when the TKST bit is set from low to high. When the TKBUSY bit is changed from high to low, the M0KOEN bit will automatically be cleared to zero to disable the key oscillator.

In the manual scan mode, the key oscillator should first be enabled before setting the TKST bit from low to high if the relevant key is enabled to be scanned and will be disabled when the TKBUSY bit is changed from high to low.

- Bit 3 **M0K4EN**: Touch key module 0 KEY4 enable control
 0: Disable
 1: Enable
- Bit 2 **M0K3EN**: Touch key module 0 KEY3 enable control
 0: Disable
 1: Enable
- Bit 1 **M0K2EN**: Touch key module 0 KEY2 enable control
 0: Disable
 1: Enable
- Bit 0 **M0K1EN**: Touch key module 0 KEY1 enable control
 0: Disable
 1: Enable

• **TKM0C2 Register**

This register is used to select the desired scan key in the time slots 0~3 of the touch key module 0. It should be noted that if any key is disabled, the touch key module 0 Reference and Key oscillators of the corresponding time slot will not oscillate.

Bit	7	6	5	4	3	2	1	0
Name	M0SK31	M0SK30	M0SK21	M0SK20	M0SK11	M0SK10	M0SK01	M0SK00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	1	0	0

- Bit 7~6 **M0SK31~M0SK30**: Touch key module 0 time slot 3 key scan selection
 00: KEY1
 01: KEY2
 10: KEY3
 11: KEY4

These bits are used to select the desired scan key in time slot 3. The settings for these bits are invalid when TKMOD1~TKMOD0=01 or TSC=1.

- Bit 5~4 **M0SK21~M0SK20**: Touch key module 0 time slot 2 key scan selection
 00: KEY1
 01: KEY2
 10: KEY3
 11: KEY4

These bits are used to select the desired scan key in time slot 2. The settings for these bits are invalid when TKMOD1~TKMOD0=01 or TSC=1.

- Bit 3~2 **M0SK11~M0SK10**: Touch key module 0 time slot 1 key scan selection
 00: KEY1
 01: KEY2
 10: KEY3
 11: KEY4

These bits are used to select the desired scan key in time slot 1. The settings for these bits are invalid when TKMOD1~TKMOD0=01 or TSC=1.

- Bit 1~0 **M0SK01~M0SK00**: Touch key module 0 time slot 0 key scan selection
 00: KEY1
 01: KEY2
 10: KEY3
 11: KEY4

These bits are used to select the desired scan key in time slot 0 in the auto scan mode or the periodic auto scan mode or used as the multiplexer for scan key select in the manual mode.

• **TKM0TH16H/TKM0TH16L – Touch Key Module 0 16-bit Threshold Register Pair**

Register	TKM0TH16H								TKM0TH16L							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/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	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register pair is used to store the touch key module 0 16-bit upper/lower threshold value. This register pair will be loaded with the corresponding upper/lower threshold value from the dedicated touch key data memory automatically by the hardware before scanning the touch key. After the touch key module 0 dedicated touch key, KEY_n (n=1~4), scan operation is completed, the 16-bit C/F counter content, TKM016DH/TKM016DL, will be compared with the TKM0TH16H/TKM0TH16L value by the hardware. When this value is less than the lower threshold if M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1, then the M0KnTHF flag will be set high, and an interrupt signal will be generated.

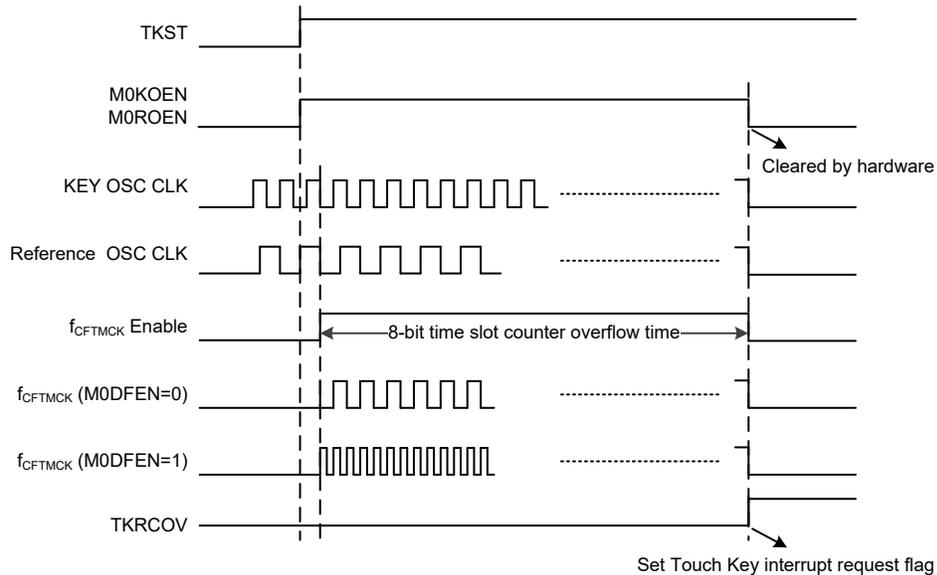
• **TKM0THS Register**

Bit	7	6	5	4	3	2	1	0
Name	M0K4THF	M0K3THF	M0K2THF	M0K1THF	M0K4THS	M0K3THS	M0K2THS	M0K1THS
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 **M0K4THF**: Touch key module 0 KEY4 upper/lower threshold comparison flag
 0: Not less than lower threshold or not larger than upper threshold
 1: Less than lower threshold or larger than upper threshold
- Bit 6 **M0K3THF**: Touch key module 0 KEY3 upper/lower threshold comparison flag
 0: Not less than lower threshold or not larger than upper threshold
 1: Less than lower threshold or larger than upper threshold
- Bit 5 **M0K2THF**: Touch key module 0 KEY2 upper/lower threshold comparison flag
 0: Not less than lower threshold or not larger than upper threshold
 1: Less than lower threshold or larger than upper threshold
- Bit 4 **M0K1THF**: Touch key module 0 KEY1 upper/lower threshold comparison flag
 0: Not less than lower threshold or not larger than upper threshold
 1: Less than lower threshold or larger than upper threshold
- Bit 3 **M0K4THS**: Touch key module 0 KEY4 upper or lower threshold comparison selection
 0: Lower threshold comparison
 1: Upper threshold comparison
- Bit 2 **M0K3THS**: Touch key module 0 KEY3 upper or lower threshold comparison selection
 0: Lower threshold comparison
 1: Upper threshold comparison
- Bit 1 **M0K2THS**: Touch key module 0 KEY2 upper or lower threshold comparison selection
 0: Lower threshold comparison
 1: Upper threshold comparison
- Bit 0 **M0K1THS**: Touch key module 0 KEY1 upper or lower threshold comparison selection
 0: Lower threshold comparison
 1: Upper threshold comparison

Touch Key Operation

When a finger touches or is in proximity to a touch pad, the capacitance of the pad will increase. By using this capacitance variation to change slightly the frequency of the internal sense oscillator, touch actions can be sensed by measuring these frequency changes. Using an internal programmable divider the reference clock is used to generate a fixed time period. By counting a number of generated clock cycles from the sense oscillator during this fixed time period touch key actions can be determined.



Touch Key Manual Scan Mode Timing Diagram

The touch key module 0 contains four touch key inputs, namely KEY1~KEY4, which are shared with logical I/O pins, and the desired function is selected using register bits. The touch key has its own independent sense oscillator. There are therefore four sense oscillators within the touch key module 0.

During this reference clock fixed interval, the number of clock cycles generated by the sense oscillator is measured, and it is this value that is used to determine if a touch action has been made or not. At the end of the fixed reference clock time interval a Touch Key TKRCOV interrupt signal will be generated in the manual scan mode.

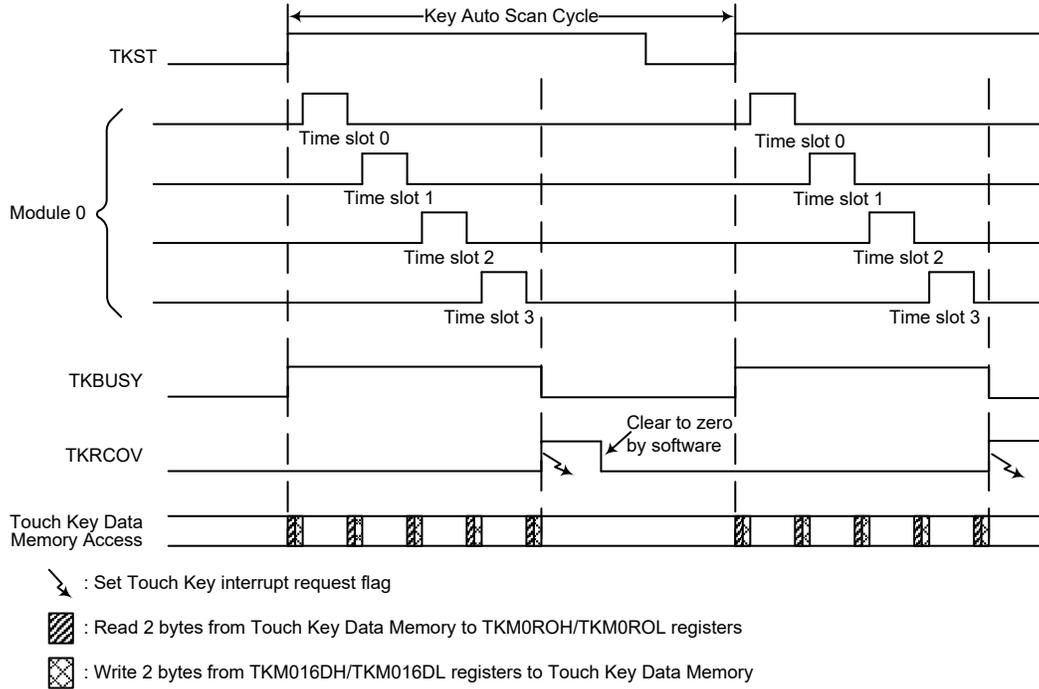
The touch key module 0 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter in the module will be automatically cleared when the TKST bit is cleared to zero, but the 8-bit programmable time slot counter will not be cleared. The overflow time is setup by user. When the TKST bit changes from low to high, the 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched on.

The key oscillator and reference oscillator in the module 0 will be automatically stopped and the 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched off when the time slot counter overflows. The clock source for the time slot counter is sourced from the reference oscillator, f_{LIRC} which is selected using the M0TSS bit in the TKM0C1 register. The reference oscillator and key oscillator will be enabled by setting the M0ROEN bit and M0KOEN bits in the TKM0C1 register.

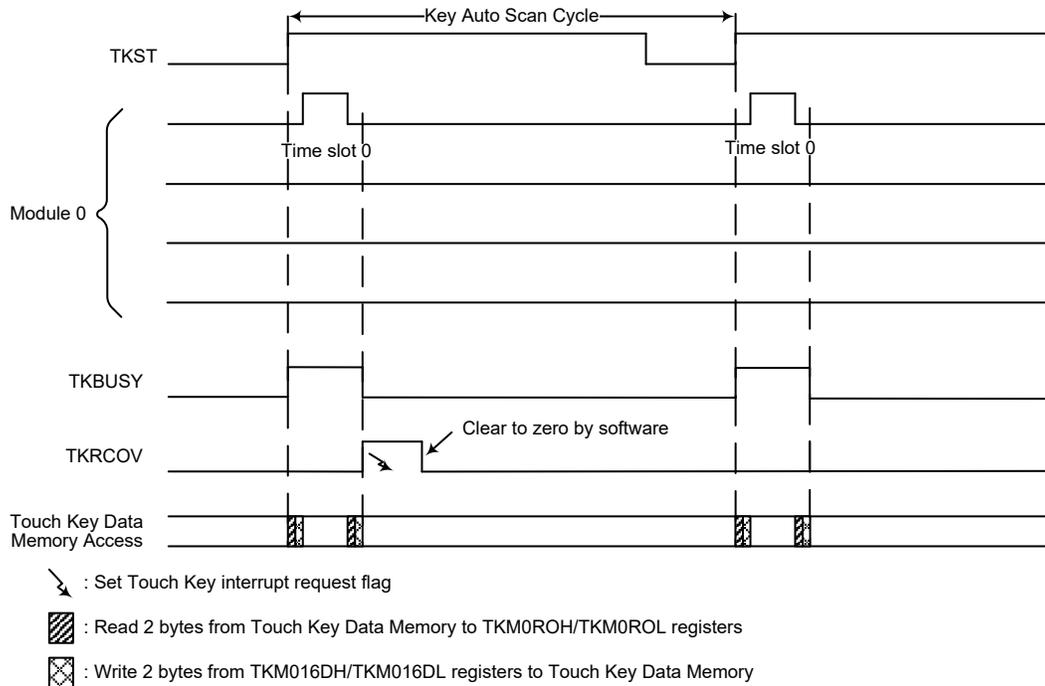
When the time slot counter in the touch key module 0 overflows, an actual touch key TKRCOV interrupt will take place. The touch keys mentioned here are the keys which are enabled.

Auto Scan Mode

There are three scan modes contained for the touch key function. The auto scan mode, the periodic auto scan mode and the manual scan mode are selected using the TKMOD1~TKMOD0 bits in the TKC0 register. The auto scan mode can minimize the load of the application program and improve the touch key scan operation performance. When the TKMOD1~TKMOD0 bits are set to 00, the auto scan mode is selected to scan the module keys in a specific sequence determined by the M0SK3[1:0]~M0SK0[1:0] bits in the TKM0C2 register. The TSC bit in the TKC2 register is used to configure time slot.



Touch Key Auto Scan Mode Timing Diagram – TSC=0



Touch Key Auto Scan Mode Timing Diagram – TSC=1

In the auto scan mode the key oscillator and reference oscillator will automatically be enabled when the TKST bit is set from low to high and disabled automatically when the TKBUSY bit changes from high to low. If the TSC bit is cleared to low, time slot 0~3 are active. When the TKST bit is set from low to high in the auto scan mode, the internal capacitor value of the reference oscillator for the selected key to be scanned in the time slot 0 will first be read from a specific location of the dedicated touch key data memory and loaded into the corresponding TKM0ROH/TKM0ROL registers. Then the 16-bit C/F counter value will be written into the corresponding location of the time slot 3 scanned key in the touch key data memory. After this, the selected key will start to be scanned in time slot 0. At the end of the time slot 0 key scan operation, the reference oscillator internal capacitor value for the next selected key will be read from the touch key data memory and loaded into the TKM0ROH/TKM0ROL registers. Then the 16-bit C/F counter value of the current scanned key will be written into the corresponding touch key data memory. The whole auto scan operation will sequentially be carried out in the above specific way from time slot 0 to time slot 3. At the end of the time slot 3 key scan operation, the reference oscillator internal capacitor value for the time slot 0 selected key will again be read from the touch key data memory and loaded into the TKM0ROH/TKM0ROL registers. Then the 16-bit C/F counter value will be written into the relevant location of the time slot 3 scanned key in the touch key data memory. After selected keys are all scanned, the TKRCOV bit will be set high and the TKBUSY bit will be cleared to zero as well as an auto scan operation is completed. If the TSC bit is set high, only time slot 0 is active, only time slot 0 is active to execute the touch key related operations and time slot 1~3 are invalid.

Periodic Auto Scan Mode

In addition to those actions mentioned in the auto scan mode, the periodic auto scan mode provides periodic auto scan and C/F counter upper/lower threshold comparison functions. When the TKMOD1~TKMOD0 bits are set to 10 or 11, the periodic auto scan mode is selected to scan the module keys automatically and periodically. Note that this mode is generally used in the IDLE mode, in order to monitor the touch key state and minimise power consumption.

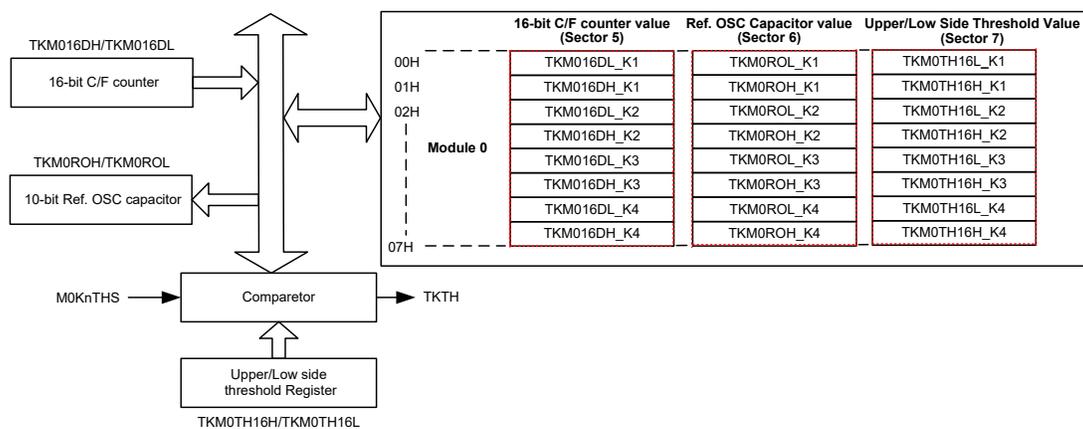
In the periodic auto scan mode the touch key scan operation will be implemented automatically on a periodic basis, which can be determined by the ASMP1~ASMP0 bits in the TKC2 register. The number of touch key scan times depends upon the WDT time-out period and the periodic auto scan mode period. Each auto scan operation will sequentially be carried out in a specific way from time slot 0 to time slot 3 like the auto scan mode. The reference oscillator internal capacitor value for each time slot selected key will be read from the touch key data memory and loaded into the TKM0ROH/TKM0ROL registers. However, only at the end of the last scan operation in the WDT time-out cycle, the 16-bit C/F counter value for all scanned keys will be written into the corresponding touch key data memory.

In addition, the 16 bit upper/lower threshold value for the selected key to be scanned in the time slot 0 will be read from a specific location of the dedicated touch key data memory and loaded into the corresponding TKM0TH16H/TKM0TH16L registers before the selected key will start to be scanned in time slot 0. The TKM0TH16H/TKM0TH16L register pair will be loaded with the corresponding next time slot 16-bit upper/lower threshold value from the dedicated touch key data memory automatically by the hardware at the end of the current time slot when the periodic auto scan mode is selected. each touch key has its own independent upper/lower threshold comparator. The upper/lower threshold comparison function will automatically be enabled in the periodic auto scan mode. When any key C/F counter value is less than the lower threshold if M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1, this indicates that the touch key state changes, then the M0KnTHF flag will be set high by the hardware, and an interrupt signal will be generated. Note that if the touch key threshold TKTH interrupt occurs, 1-byte data will be written to the TKM0ROL register because the TKM0ROH/TKM0ROL register pair will be loaded with the corresponding next time slot capacitor value from the dedicated touch key data memory and the 16-bit C/F counter content, TKM016DH/TKM016DL, and the TKM0TH16H/TKM0TH16L value are compared at the same time.

As the periodic auto scan operation is implemented using the WDT counter clock to reduce power consumption, when the WDT is cleared the WDT counter will be reset, the periodic auto scan operation time will be affected but the number of touch key times will not be affected.

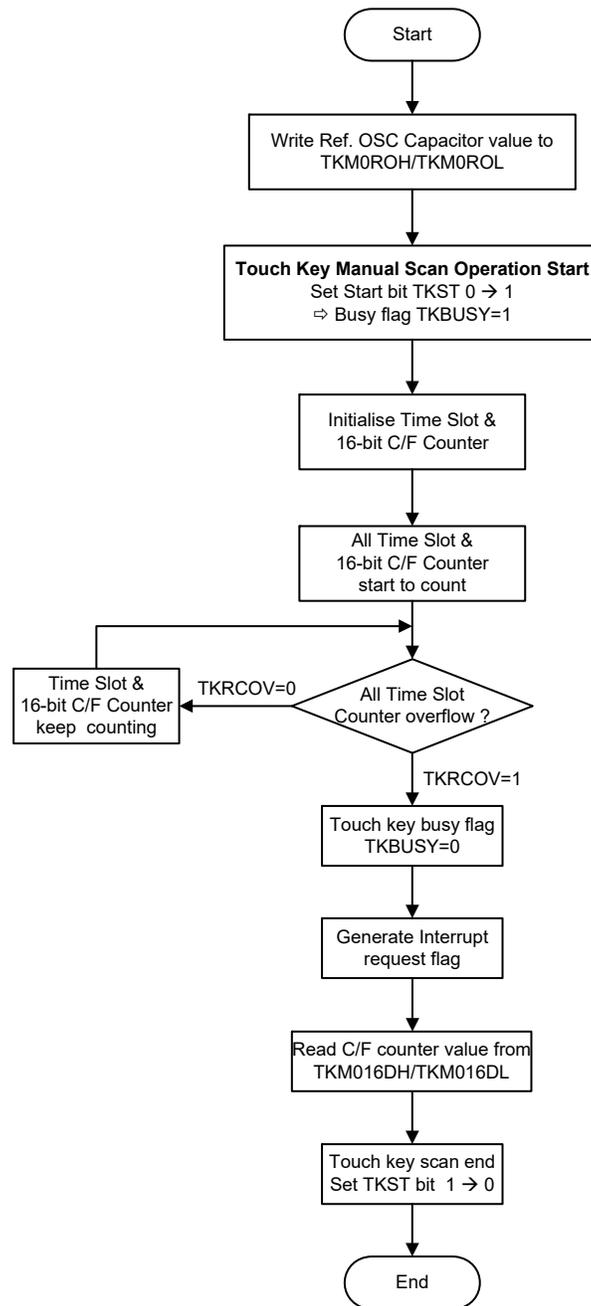
Touch Key Data Memory

The device provides three dedicated Data Memory area. The first area is used to store the 16-bit C/F counter values of the touch key module and located in Data Memory Sector 5. The second area is used to store the reference oscillator internal capacitor values of the touch key module and located in Data Memory Sector 6. The last area is used to store the 16-bit upper/lower threshold value of the touch key module and located in Data Memory Sector 7.

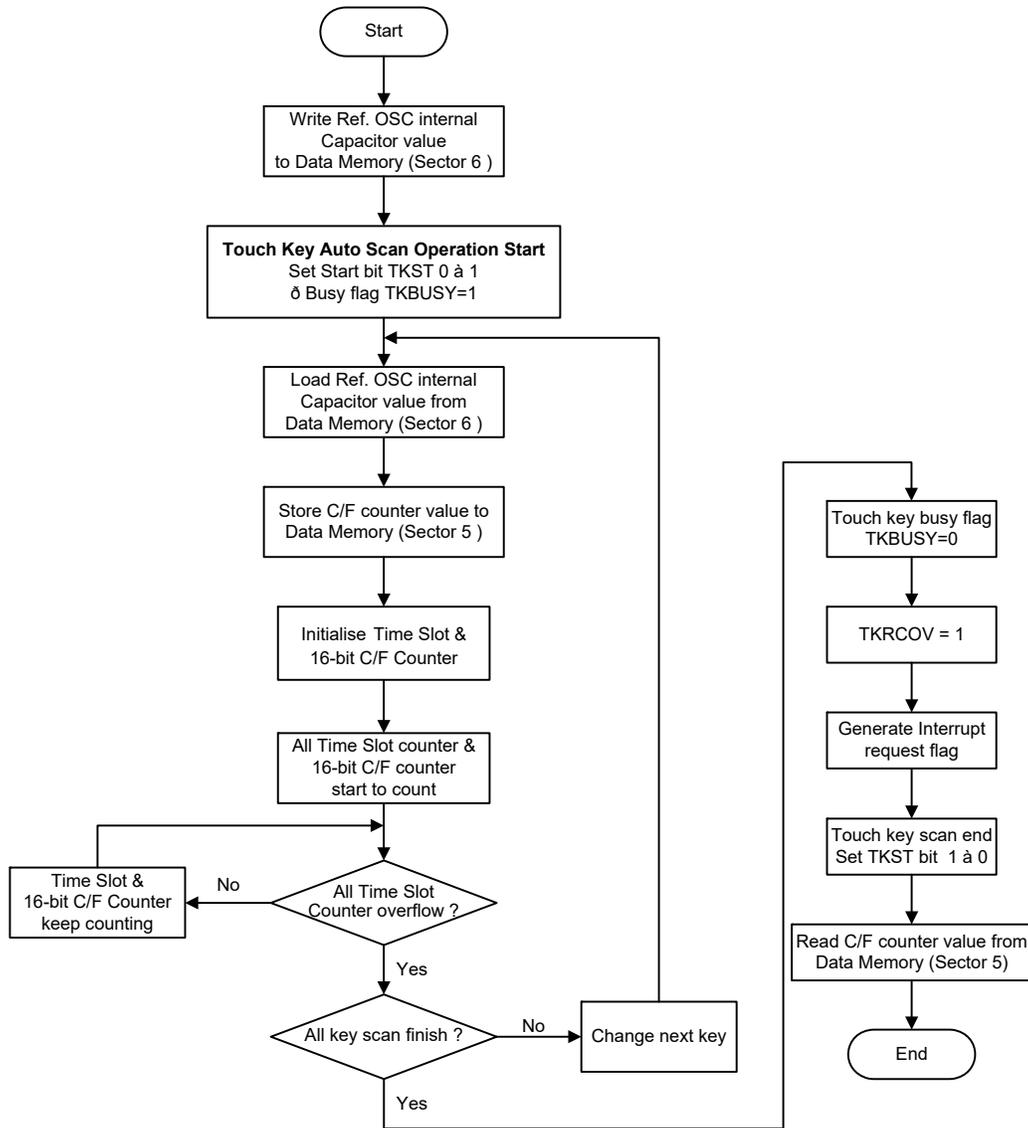


Touch Key Register Memory Map

Touch Key Scan Operation Flowchart



Touch Key Manual Scan Mode Flowchart



Touch Key Auto Scan Mode Flowchart

Touch Key Interrupts

The touch key has two independent interrupts, known as touch key TKRCOV interrupt and touch key threshold TKTH interrupt. In the manual scan mode, when the touch key module 0 time slot counter overflows, an actual touch key TKRCOV interrupt will take place. In the auto scan mode, when the touch key auto scan operation is completed, the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set. In the periodic auto scan mode, after the last scan operation in the WDT time-out cycle completes the 16-bit C/F counter content is written into the corresponding touch key data memory, then the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set. The touch keys mentioned here are the keys which are enabled. The 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter in the module will be automatically cleared. When any key C/F counter value is less than the lower threshold if M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1, a touch key threshold interrupt will take place. More details regarding the touch key interrupt is located in the interrupt section of the datasheet.

Programming Considerations

After the relevant registers are setup, the touch key detection process is initiated when changing the TKST Bit from low to high. This will enable and synchronise all relevant oscillators. The TKRCOV flag which is the time slot counter flag will go high when the counter overflows in the manual scan mode. When this happens an interrupt signal will be generated. In the auto scan mode, if the time slot counter overflows but the touch key auto scan operation is not completed, the TKRCOV bit will not be set. When the touch key auto scan operation is completed, the TKRCOV bit and the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set. In the periodic auto scan mode, the TKRCOV bit is cleared to zero during the auto scan operation period. Only at the end of the last scan operation in the WDT time-out cycle, the 16-bit C/F counter content will be written into the corresponding touch key data memory, and then the TKRCOV bit will be set high by the hardware circuit. The TKTH signal which is the threshold comparison indication signal will go high when a certain threshold comparison condition occurs. When this happens an interrupt signal will also be generated. As the TKRCOV flag will not be automatically cleared, it has to be cleared by the application program.

The TKCFOV flag which is the 16-bit C/F counter overflow flag will go high when any of the Touch Key Module 0 16-bit C/F counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program.

The TK16OV flag which is the 16-bit counter overflow flag will go high when the 16-bit counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program.

When the external touch key size and layout are defined, their related capacitances will then determine the sensor oscillator frequency.

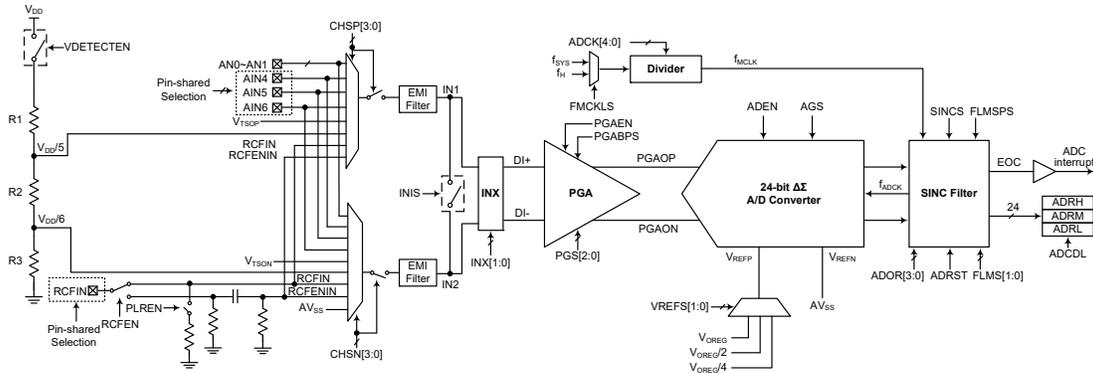
Analog to Digital Converter

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

A/D Converter Overview

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

In addition, PGA gain control and A/D converter gain control determine the amplification gain for A/D converter input signal. The designer can select the best gain combination for the desired amplification applied to the input signal. The following block diagram illustrates the A/D converter basic operational function. The A/D converter external input channel can be arranged as 6 single-ended A/D converter input channels and 3 differential input channels. The input signal can be amplified by PGA before entering the 24-bit Delta Sigma A/D converter. The A/D converter module will output one bit converted data to SINC filter which can transform the converted one-bit data to 24 bits and store them into the specific data registers. Additionally, the device also provides a temperature sensor to compensate the A/D converter deviation caused by the temperature. With high accuracy and performance, the device is very suitable for differential output sensor applications such as weight measurement scales and other related products.

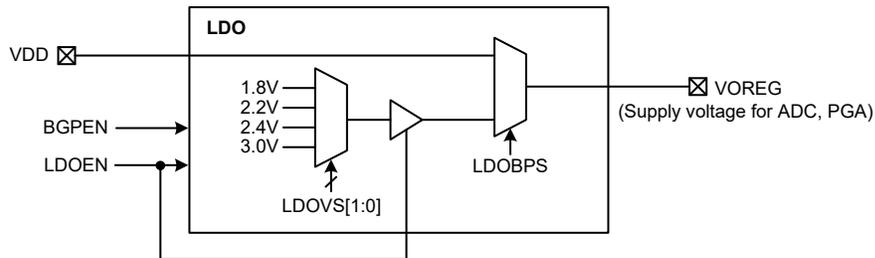


Note: The PGA and A/D converter are power supplied by the V_{OREG} or V_{DD}, which is selected by the VADCS bit in the PGAC1 register.

A/D Converter Structure

Internal Power Supply

The device contains an LDO for the regulated power supply. The accompanying block diagram illustrates the basic functional operation. The internal LDO can provide the fixed voltage for the PGA, A/D converter or the external components. There are four LDO voltage levels, 1.8V, 2.2V, 2.4V or 3.0V, determined by the LDOVS1~LDOVS0 bits in the PWRC register. The LDO function can be controlled by the LDOEN bit and can be powered off to reduce overall power consumption. When the LDO function is enabled, the BGPEN bit should be set high to provide the LDO reference voltage.



Internal Power Supply Block Diagram

• PWRC Register

Bit	7	6	5	4	3	2	1	0
Name	LDOEN	BGPEN	CLK_EN	VDTECTEN	TSEN	LDOBPS	LDOVS1	LDOVS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	1	0

Bit 7 **LDOEN**: LDO function enable control
0: Disable
1: Enable

If the LDO is disabled, there will be no power consumption and the LDO output will be in a low level using a weak internal pull-low resistor.

Bit 6 **BGPEN**: Bandgap enable control
0: Disable
1: Enable

When the LDO function is enabled, the BGPEN bit should be set high to provide the LDO reference voltage.

Bit 5	<p>CLK_EN: Temperature sensor clock enable control 0: Disable 1: Enable</p> <p>To start up the temperature sensor successfully, the setting sequence of the TSEN and CLK_EN bits is “00→10→11”.</p>
Bit 4	<p>VDETECTEN: Voltage divider V_{DD}/5 and V_{DD}/6 enable control 0: Disable 1: Enable</p>
Bit 3	<p>TSEN: Internal temperature sensor enable control 0: Disable 1: Enable</p>
Bit 2	<p>LDOBPS: LDO bypass enable control 0: Disable 1: Enable</p>
Bit 1~0	<p>LDOVS1~LDOVS0: LDO output voltage selection 00: 1.8V 01: 2.2V 10: 2.4V 11: 3.0V</p>

A/D Converter Data Rate Definition

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

Data Rate for SINC2

$$\begin{aligned}
 &= f_{ADCK} / (2 \times OSR) \\
 &= (f_{MCLK} / N) / (2 \times OSR) \\
 &= f_{MCLK} / (N \times 2 \times OSR)
 \end{aligned}$$

Data Rate for SINC3

$$\begin{aligned}
 &= f_{ADCK} / OSR \\
 &= (f_{MCLK} / N) / OSR \\
 &= f_{MCLK} / (N \times OSR)
 \end{aligned}$$

f_{ADCK} : A/D converter clock frequency, derived from f_{MCLK}/N .

f_{MCLK} : A/D converter clock source, derived from f_{SYS} or f_H or f_{SYS} (or f_H)/2/(ADCK[4:0]+1) using the ADCK bit field.

N: A constant divide factor equal to 12 or 30 determined by the FLMS bit field.

OSR: Oversampling rate determined by the ADOR bit field.

For example, if a data rate of 8Hz is desired, an f_{MCLK} clock source with a frequency of 4MHz A/D converter can be selected. Then set the FLMS field to “00” to obtain an “N” equal to 30.

For the SINC2 filter, set the ADOR field to “0010” to select an oversampling rate equal to 8192. Therefore, the Data Rate=4MHz/(30×2×8192)=8Hz.

For the SINC3 filter, set the ADOR field to “0001” to select an oversampling rate equal to 16384. Therefore, the Data Rate=4MHz/(30×16384)=8Hz.

A/D Converter Register Description

Overall operation of the A/D converter is controlled by using a series of registers. A control register named as PWRC is used to control the power supply related settings for the PGA and A/D converter. This register is described in the “Internal Power Supply” section. Three read only registers exist to store the A/D converter data 24-bit value. The remaining registers are control registers which setup the gain selections and control functions of the PGA and A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PWRC	LDOEN	BGPEN	CLK_EN	VDETECTEN	TSEN	LDOBPS	LDOVS1	LDOVS0
PGAC0	VREFS1	VREFS0	AGS	PGAEN	PGABPS	PGS2	PGS1	PGS0
PGAC1	INIS	INX1	INX0	—	—	PLREN	VADCS	RCFEN
PGACS	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
ADRL	D7	D6	D5	D4	D3	D2	D1	D0
ADRM	D15	D14	D13	D12	D11	D10	D9	D8
ADRH	D23	D22	D21	D20	D19	D18	D17	D16
ADCR0	SINCS	D6	FLMS1	FLMS0	ADOR3	ADOR2	ADOR1	ADOR0
ADCR1	—	—	ADCDL	EOC	ADRST	ADEN	—	—
ADCS	FLMSPS	FMCLKS	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0

A/D Converter Register List

Programmable Gain Amplifier Registers – PGAC0, PGAC1, PGACS

There are three registers related to the programmable gain control, PGAC0, PGAC1 and PGACS. The PGAC0 register is used to select the PGA gain, the A/D Converter gain and the A/D converter reference voltage pair as well as control the PGA enable/disable and PGA bypass function. The PGAC1 register is used to define the input connection, control the pull-low resistor and RC filter enable/disable and select the A/D converter voltage. In addition, the PGACS register is used to select the PGA inputs. Therefore, the input channels have to be determined by the CHSP3~CHSP0 and CHSN3~CHSN0 bits to determine which analog channel input pins, temperature sensor inputs or internal power supply are actually connected to the internal differential A/D converter.

• PGAC0 Register

Bit	7	6	5	4	3	2	1	0
Name	VREFS1	VREFS0	AGS	PGAEN	PGABPS	PGS2	PGS1	PGS0
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 **VREFS1~VREFS0**: A/D converter reference voltage pair selection

- 00: Internal reference voltage pair – V_{OREG} & AV_{SS}
- 01: Internal reference voltage pair – V_{OREG} & AV_{SS}
- 10: Internal reference voltage pair – $V_{OREG}/2$ & AV_{SS}
- 11: Internal reference voltage pair – $V_{OREG}/4$ & AV_{SS}

Bit 5 **AGS**: A/D converter PGAOP/PGAON differential input signal gain selection

- 0: ADGN=1
- 1: ADGN=2

Bit 4 **PGAEN**: PGA enable control

- 0: Disable
- 1: Enable

This bit controls the overall enable/disable function of the PGA circuits. The related functions will only be available when the PGAEN is set high.

Bit 3 **PGABPS**: PGA bypass control

- 0: Normal mode
- 1: Bypass PGA mode

Bit 2~0 **PGS2~PGS0**: PGA DI+/DI- differential channel input gain selection

- 000: PGAGN=1
- 001: PGAGN=2
- 010: PGAGN=4
- 011: PGAGN=8

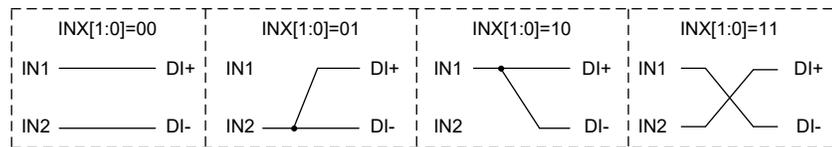
100: PGAGN=16
 101: PGAGN=32
 110: PGAGN=64
 111: PGAGN=128

• **PGAC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	INIS	INX1	INX0	—	—	PLREN	VADCS	RCFEN
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7 **INIS**: The selected input ends, IN1/IN2, internal connection control
 0: Not connected
 1: Connected

Bit 6~5 **INX1~INX0**: The selected input ends, IN1/IN2, and the PGA differential input ends, DI+/DI-, connection control



Bit 4~3 Unimplemented, read as “0”

Bit 2 **PLREN**: Pull-low resistor enable control
 0: Disable
 1: Enable

Bit 1 **VADCS**: PGA and A/D converter voltage selection
 0: V_{REG}
 1: AV_{DD}

Bit 0 **RCFEN**: RC filter input enable control
 0: Disable
 1: Enable

• **PGACS Register**

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

Bit 7~4 **CHSN3~CHSN0**: Negative input end IN2 selection
 0000: AN0
 0001: AN1
 0010~0111: Reserved
 1000: AIN4
 1001: AIN5
 1010: AIN6
 1011: Temperature sensor output – V_{TSON}
 1100: V_{DD}/6
 1101: RCFIN
 1110: RCFENIN
 1111: AV_{SS}

These bits are used to select the negative input, IN2. It is recommended that when the V_{TSON} signal is selected as the negative input, the V_{TSON} signal should be selected as the positive input for proper operations.

- Bit 3~0 **CHSP3~CHSP0:** Positive input end IN1 selection
 0000: AN0
 0001: AN1
 0010~0111: Reserved
 1000: AIN4
 1001: AIN5
 1010: AIN6
 1011: Temperature sensor output – V_{TSOP}
 1100: $V_{DD}/5$
 1101: RCFIN
 1110: RCFENIN
 1111: Reserved

These bits are used to select the positive input, IN1. It is recommended that when the V_{TSOP} signal is selected as the positive input, the V_{TSOP} signal should be selected as the negative input for proper operations.

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

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

• ADRL Register

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

“x”: unknown

Bit 7~0 **D7~D0:** A/D conversion data register bit 7 ~ bit 0

• ADRM Register

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

“x”: unknown

Bit 7~0 **D15~D8:** A/D conversion data register bit 15 ~ bit 8

• ADRH Register

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

“x”: unknown

Bit 7~0 **D23~D16:** A/D conversion data register bit 23 ~ bit 16

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

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

• **ADCR0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SINCS	D6	FLMS1	FLMS0	ADOR3	ADOR2	ADOR1	ADOR0
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 **SINCS**: SINC selection
 0: SINC3 filter
 1: SINC2 filter
- Bit 6 **D6**: Reserved, must be fixed at “0”
- Bit 5~4 **FLMS1~FLMS0**: A/D converter clock f_{ADCK} selection
 00: $f_{ADCK}=f_{MCLK}/30, N=30$
 10: $f_{ADCK}=f_{MCLK}/12, N=12$
 Others: Reserved
- Bit 3~0 **ADOR3~ADOR0**: A/D conversion oversampling rate selection
 0000: Oversampling rate OSR=32768
 0001: Oversampling rate OSR=16384
 0010: Oversampling rate OSR=8192
 0011: Oversampling rate OSR=4096
 0100: Oversampling rate OSR=2048
 0101: Oversampling rate OSR=1024
 0110: Oversampling rate OSR=512
 0111: Oversampling rate OSR=256
 1000: Oversampling rate OSR=128
 1001: Oversampling rate OSR=64
 1010: Oversampling rate OSR=32
 1011: Oversampling rate OSR=16
 1100: Oversampling rate OSR=8
 Others: Reserved

• **ADCR1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	ADCDL	EOC	ADRST	ADEN	—	—
R/W	—	—	R/W	R/W	R/W	R/W	—	—
POR	—	—	0	0	0	0	—	—

- Bit 7~6 Unimplemented, read as “0”
- Bit 5 **ADCDL**: A/D converted data latch function enable control
 0: Disable
 1: Enable

If the A/D converted data latch function is enabled, the latest converted data value will be latched and will not be updated by any subsequent converted results until this function is disabled. Although the converted data is latched into the data registers, the A/D converter circuits remain operational, but will not generate an interrupt and the EOC will not change. It is recommended that this bit should be set high before reading the converted data from the ADRL, ADRM and ADRH registers. After the converted data has been read out, the bit can then be cleared to zero to disable the A/D converter data latch function and allow further conversion values to be stored. In this way,

the possibility of obtaining undesired data during A/D converter conversions can be prevented.

Bit 4 **EOC**: End of A/D conversion flag
 0: A/D conversion in progress
 1: A/D conversion ended
 This bit must be cleared by software.

Bit 3 **ADRST**: A/D converter software reset enable control
 0: Disable
 1: Enable
 This bit is used to reset the A/D converter internal digital SINC filter. This bit is cleared to zero for A/D normal operations. However, if set high, the internal digital SINC filter will be reset and the current A/D converted data will be aborted. A new A/D data conversion process will not be initiated until this bit is cleared to zero again.

Bit 2 **ADEN**: A/D converter function enable control
 0: Disable
 1: Enable
 This bit controls the A/D converter function enable/disable. This bit should be set high 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. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.

Bit 1~0 Unimplemented, read as “0”

• **ADCS Register**

Bit	7	6	5	4	3	2	1	0
Name	FLMSPS	FMCLKS	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
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 **FLMSPS**: FLMS1~FLMS0 A/D converter clock division bypass control
 0: Disable
 1: Enable

Bit 6 **FMCLKS**: f_{MCLK} clock source selection
 0: f_{SYS}
 1: f_H

Bit 5 Unimplemented, read as “0”

Bit 4~0 **ADCK4~ADCK0**: A/D converter clock source f_{MCLK} setup
 00000~11110: $f_{MCLK}=f_{SYS}$ (or f_H)/2/(ADCK[4:0]+1)
 11111: $f_{MCLK}=f_{SYS}$ (or f_H)

A/D Converter Operation

The ADRST bit in the ADCR1 register is used to start and reset the A/D converter after power on. When the microcontroller changes this bit from low to high and then low again, an analog to digital conversion in the SINC filter will be initiated. After this setup is completed, the A/D Converter is ready for operation. This bit is used to control the overall start operation of the internal analog to digital converter.

The EOC bit in the ADCR1 register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set high by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D converter interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D converter internal interrupt signal will direct the program flow to the associated A/D converter internal interrupt address for processing. If the A/D converter internal

interrupt is disabled, the microcontroller can poll the EOC bit in the ADCR1 register to check whether it has been set to “1” as an alternative method of detecting the end of an A/D conversion cycle. The A/D converted data will be updated continuously by the new converted data. If the A/D converted data latch function is enabled, the latest converted data will be latched and the following new converted data will be discarded until this data latch function is disabled.

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

The differential reference voltage supply to the A/D Converter can be supplied from V_{OREG} and AV_{SS} , or from a subdivided version of V_{OREG} and AV_{SS} . The desired selection is made using the VREFS1~VREFS0 bits in the PGAC0 register.

Summary of A/D Conversion Steps

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

- Step 1
Select the PGA and A/D converter power supply by the VADCS bit in the PGAC1 register.
- Step 2
Select the PGA gain and the A/D converter gain as well as the A/D converter reference voltage pair by the PGAC0 register.
- Step 3
Select the PGA input connection by the PGAC1 register.
- Step 4
Select the required A/D conversion clock source by correctly programming the ADCK4~ADCK0 and FLMSPS bits in the ADCS register.
- Step 5
Select output data rate by configuring the ADOR2~ADOR0 and FLMS1~FLMS0 bits in the ADCR0 register. If the FLMSPS bit in the ADCS register is set high, the FLMS1~FLMS0 bits can not be configured.
- Step 6
Select which channel is to be connected to the internal PGA by correctly programming the CHSP3~CHSP0 and CHSN3~CHSN0 bits which are also contained in the PGACS register.
- Step 7
Enable the A/D converter by setting the ADEN bit in the ADCR1 register high.
- Step 8
Reset the A/D converter by setting the ADRST bit in the ADCR1 register high and then clearing this bit to zero to release the reset status.
- Step 9
If the A/D converter interrupt is to be used, the related interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bit, ADE, must both be set high to do this.

- Step 10

To check when the analog to digital conversion process is complete, the EOC bit in the ADCR1 register can be polled. The conversion process is complete when this bit goes high. When this occurs the A/D converter data registers ADRL, ADRM and ADRH can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D converter interrupt to occur.

Note: When checking for the end of the conversion process, if the method of polling the EOC bit in the ADCR1 register is used, the interrupt enable step above can be omitted.

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D converter internal circuitry can be switched off to reduce power consumption, by clearing the ADEN bit in the ADCR1 register to zero. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines.

A/D Converter Transfer Function

The device contains a 24-bit Delta Sigma A/D converter and its full-scale converted digitized value is from 8388607 to -8388608 in decimal value. The converted data format is formed using a two's complement binary value. The MSB of the converted data is the signed bit. Since the full-scale analog input value is equal to the differential reference input voltage, ΔVR_{\pm} , selected by the VREFS1~VREFS0 bits in the PGAC0 register, this gives a single bit analog input value of ΔVR_{\pm} divided by 8388608.

$$1 \text{ LSB} = \Delta VR_{\pm} / 8388608$$

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

$$\Delta SI_I = PGAGN \times ADGN \times \Delta DI_{\pm}$$

$$ADC_Conversion_Data = (\Delta SI_I / \Delta VR_{\pm}) \times K$$

Where K is equal to 2^{23}

Note: 1. The PGAGN and ADGN values are decided by the PGS[2:0] and AGS control bits

2. ΔSI_I : Differential Input Signal after amplification
3. PGAGN: Programmable Gain Amplifier gain
4. ADGN: A/D Converter gain
5. ΔDI_{\pm} : Differential input signal derived from external channels or internal signals
6. ΔVR_{\pm} : Differential Reference voltage

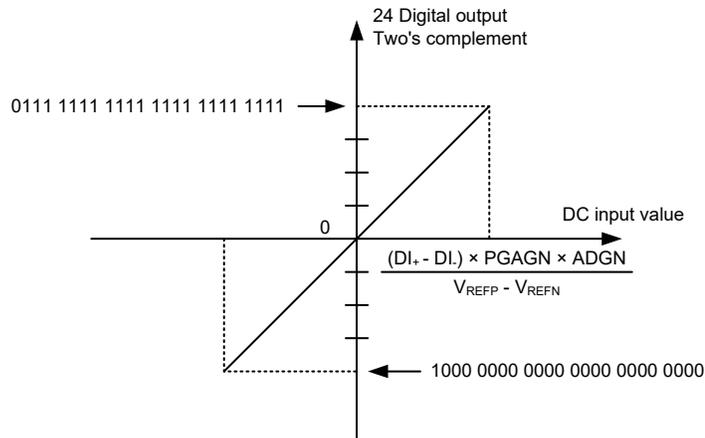
Due to the digital system design of the Delta Sigma A/D Converter, the maximum A/D converted value is 8388607 and the minimum value is -8388608. Therefore, there is a middle value of 0. The ADC_Conversion_Data equation illustrates this range of converted data variation.

A/D Conversion Data (2's complement, Hexadecimal)	Decimal Value
0x7FFFFFFF	8388607
0x800000	-8388608

A/D Conversion Data Range

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

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



A/D Converted Data

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

A/D Converted Data to Voltage

The converted data can be recovered using the following equations:

If MSB=0–Positive Converted data

$$\text{Input Voltage} = \frac{(\text{Converted_data}) \times \text{LSB}}{\text{PGAGN} \times \text{ADGN}}$$

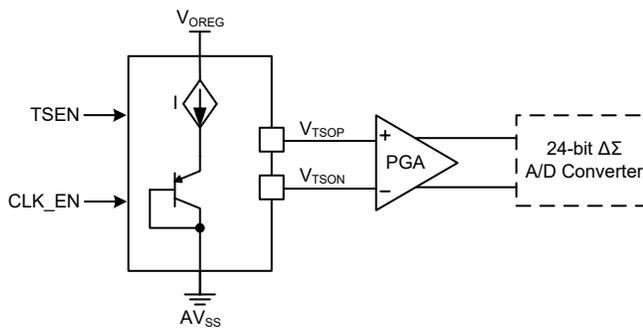
If the MSB=1–Negative Converted data

$$\text{Input voltage} = \frac{(\text{Two's_complement_of_Converted_data}) \times \text{LSB}}{\text{PGAGN} \times \text{ADGN}}$$

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

Temperature Sensor

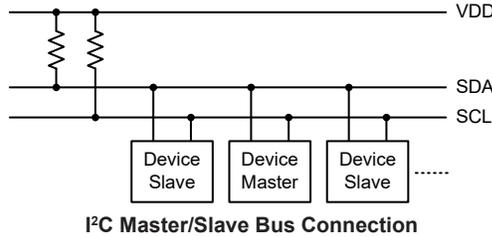
An internal temperature sensor is integrated within the device to allow compensation for temperature effects. By selecting the PGA input channels to the V_{TSOP} and V_{TSON} signals, the A/D Converter can obtain temperature information and allow compensation to be carried out on the A/D converted data. The following block diagram illustrates the functional operation for the temperature sensor.



Temperature Sensor Structure

I²C Interface

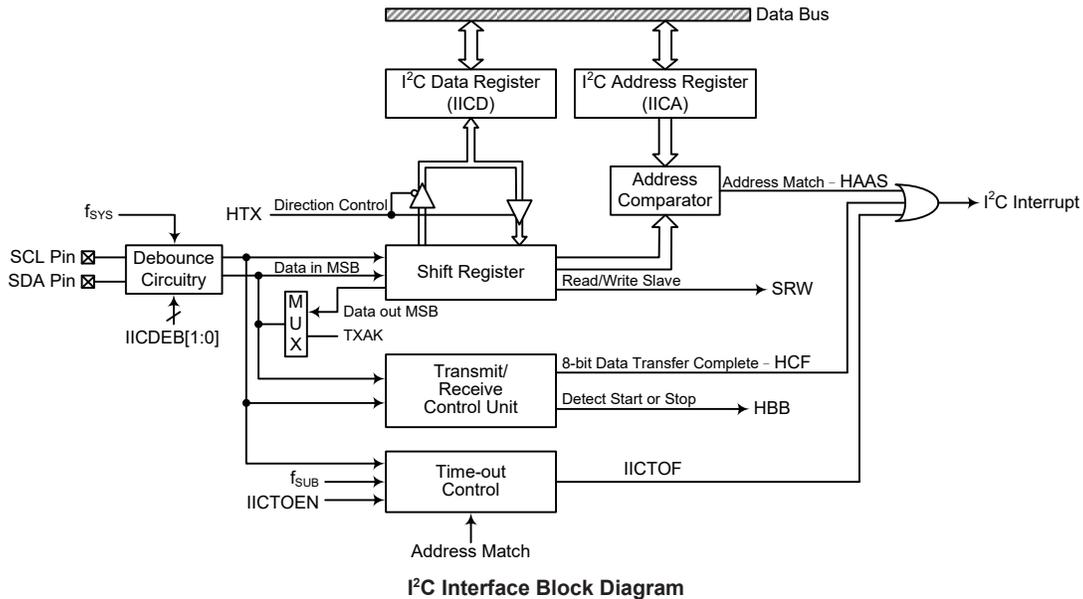
The I²C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication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

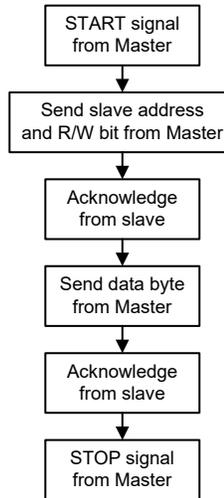


I²C Interface Operation

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

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data; however, it is the master device that has overall control of the bus. For the device, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode. It is suggested that the device should not enter the IDLE/SLEEP mode during the I²C communication.





I²C Interface Operation

The IICDEB1 and IICDEB0 bits determine the debounce time of the I²C interface. This uses the internal clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required I²C data transfer speed, there exists a relationship between the system clock, f_{SYS} , and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

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

I²C Minimum f_{SYS} Frequency Requirements

I²C Registers

There are three control registers associated with the I²C bus, IICC0, IICC1 and IICTOC, one address register IICA and one data register, IICD.

Register Name	Bit							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
IICC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
IICD	D7	D6	D5	D4	D3	D2	D1	D0
IICA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
IICTOC	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0

I²C Register List

I²C Data Register

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

• **IICD Register**

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

“x”: Unknown

Bit 7~0 **D7~D0**: I²C data register bit 7 ~ bit 0

• **I²C Address Register**

The IICA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the IICA register define the device slave address. Bit 0 is not defined. When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the IICA register, the slave device will be selected.

• **IICA Register**

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	0	0	0	0	0	0	0	—

Bit 7~1 **IICA6~IICA0**: I²C slave address
IICA6~IICA0 is the I²C slave address bit 6 ~ bit 0.

Bit 0 Unimplemented, read as “0”

• **I²C Control Registers**

There are three control registers for the I²C interface, IICC0, IICC1 and IICTOC. The IICC0 register is used to control the enable/disable function and to select the I²C mode and debounce time. The IICC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, IICTOC, is used to control the I²C time-out function and is described in the corresponding section.

• **IICC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 Unimplemented, read as “0”

Bit 3~2 **IICDEB1~IICDEB0**: I²C debounce time selection
00: No debounce
01: 2 system clock debounce
1x: 4 system clock debounce

Note that the I²C debounce circuit will operate normally if the system clock, f_{SYS} , is derived from the f_{H} clock or the IAMWU bit is equal to 0. Otherwise, the debounce circuit will have no effect and be bypassed.

Bit 1 **IICEN**: I²C enable control
0: Disable
1: Enable

The bit is the overall on/off control for the I²C interface. When the IICEN bit is cleared to zero to disable the I²C interface, the SDA and SCL lines will lose their I²C function and the I²C operating current will be reduced to a minimum value. When the bit is high the I²C interface is enabled. If the IICEN bit changes from low to high, the

contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 Unimplemented, read as “0”

• **IICC1 Register**

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

Bit 7 **HCF**: I²C bus data transfer completion flag

- 0: Data is being transferred
- 1: Completion of an 8-bit data transfer

The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated. Below is an example of the flow of a two-byte I²C data transfer.

First, I²C slave device receives a start signal from I²C master and then HCF bit is automatically cleared to zero. Second, I²C slave device finishes receiving the 1st data byte and then HCF bit is automatically set high. Third, user read the 1st data byte from IICD register by the application program and then HCF bit is automatically cleared to zero. Fourth, I²C slave device finishes receiving the 2nd data byte and then HCF bit is automatically set to one and so on. Finally, I²C slave device receives a stop signal from I²C master and then HCF bit is automatically set high.

Bit 6 **HAAS**: I²C bus address match flag

- 0: Not address match
- 1: Address match

The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB**: I²C bus busy flag

- 0: I²C Bus is not busy
- 1: I²C Bus is busy

The HBB flag is the I²C busy flag. This flag will be “1” when the I²C bus is busy which will occur when a START signal is detected. The flag will be set to “0” when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX**: I²C slave device is transmitter or receiver selection

- 0: Slave device is the receiver
- 1: Slave device is the transmitter

Bit 3 **TXAK**: I²C bus transmit acknowledge flag

- 0: Slave send acknowledge flag
- 1: Slave do not send acknowledge flag

The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8 bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set TXAK bit to “0” before further data is received.

Bit 2 **SRW**: I²C slave read/write flag

- 0: Slave device should be in receive mode
- 1: Slave device should be in transmit mode

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

bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

Bit 1 **IAMWU**: I²C address match wake-up control
 0: Disable
 1: Enable

This bit should be set to 1 to enable the I²C address match wake-up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake-up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

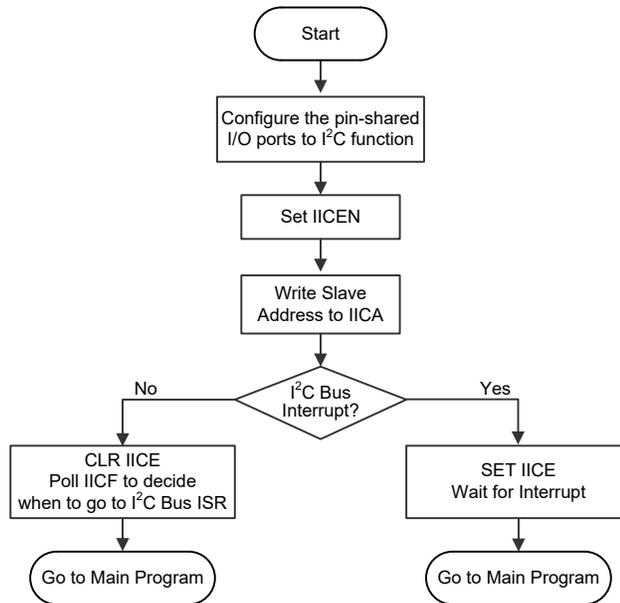
Bit 0 **RXAK**: I²C bus receive acknowledge flag
 0: Slave receive acknowledge flag
 1: Slave does not receive acknowledge flag

The RXAK flag is the receiver acknowledge flag. When the RXAK flag is “0”, it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is “1”. When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

I²C Bus Communication

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the IICC1 register will be set and an I²C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and IICTOF bits to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer completion or from the I²C bus time-out occurrence. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
 Configure the corresponding pin-shared function as the I²C function pins and set the IICEN bit to “1” to enable the I²C bus.
- Step 2
 Write the slave address of the device to the I²C bus address register IICA.
- Step 3
 Set the I²C interrupt enable bit of the interrupt control register to enable the I²C interrupt.



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

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

I²C Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the IICC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

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

I²C Bus Read/Write Signal

The SRW bit in the IICC1 register defines whether the master device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is “1” then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as

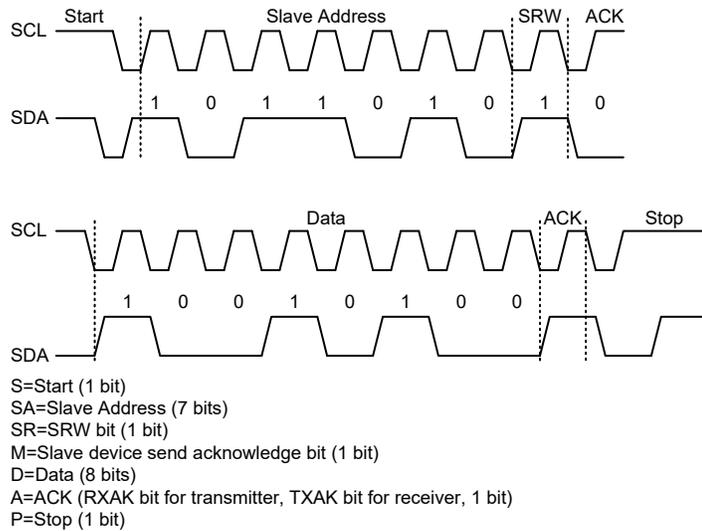
a transmitter. If the SRW flag is “0” then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the IICC1 register should be set to “1”. If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the IICC1 register should be set to “0”.

I²C Bus Data and Acknowledge Signal

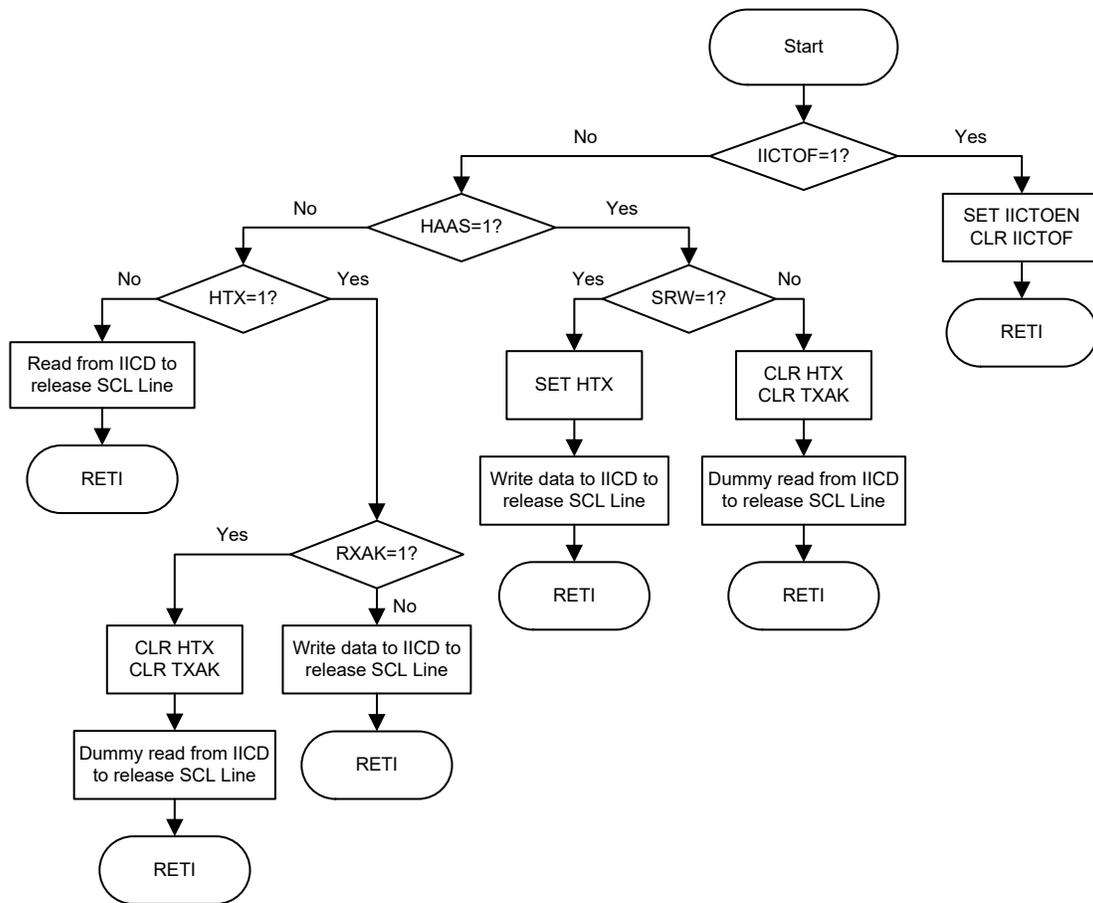
The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level “0”, before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the IICD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the IICD register. If setup as a receiver, the slave device must read the transmitted data from the IICD register. When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the RXAK bit in the IICC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.



S	SA	SR	M	D	A	D	A	S	SA	SR	M	D	A	D	A	P
---	----	----	---	---	---	---	---	-------	---	----	----	---	---	---	---	---	-------	---

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

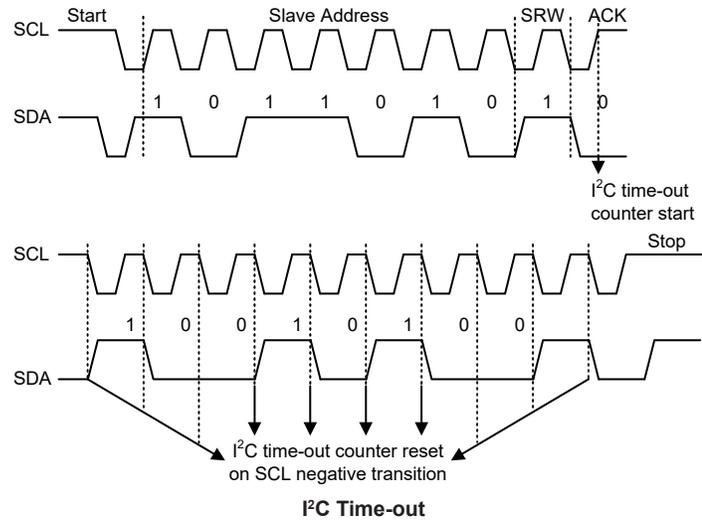
I²C Communication Timing Diagram



I²C Bus ISR Flow Chart

I²C Time-out Control

In order to reduce the problem of I²C lockup due to reception of erroneous clock sources, a time-out function is provided. If the clock source to the I²C is not received for a while, then the I²C circuitry and registers will be reset after a certain time-out period. The time-out counter starts counting on an I²C bus “START” & “address match” condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out setup by the IICTOC register, then a time-out condition will occur. The time-out function will stop when an I²C “STOP” condition occurs.



When an I²C time-out counter overflow occurs, the counter will stop and the IIC_{TOEN} bit will be cleared to zero and the IIC_{TOF} bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the I²C interrupt vector. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

Registers	After I ² C Time-out
IICD, IICA, IICC0	No change
IICC1	Reset to POR condition

I²C Registers after Time-out

The IIC_{TOF} flag can be cleared by the application program. There are 64 time-out periods which can be selected using IIC_{TOS5}~IIC_{TOS0} bits in the IIC_{TOC} register. The time-out time is given by the formula: $((1\sim64)\times(32/f_{SUB}))$. This gives a time-out period which ranges from about 1ms to 64ms.

• IIC_{TOC} Register

Bit	7	6	5	4	3	2	1	0
Name	IIC _{TOEN}	IIC _{TOF}	IIC _{TOS5}	IIC _{TOS4}	IIC _{TOS3}	IIC _{TOS2}	IIC _{TOS1}	IIC _{TOS0}
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 **IIC_{TOEN}**: I²C time-out control
0: Disable
1: Enable

Bit 6 **IIC_{TOF}**: I²C time-out flag
0: No time-out occurred
1: Time-out occurred

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

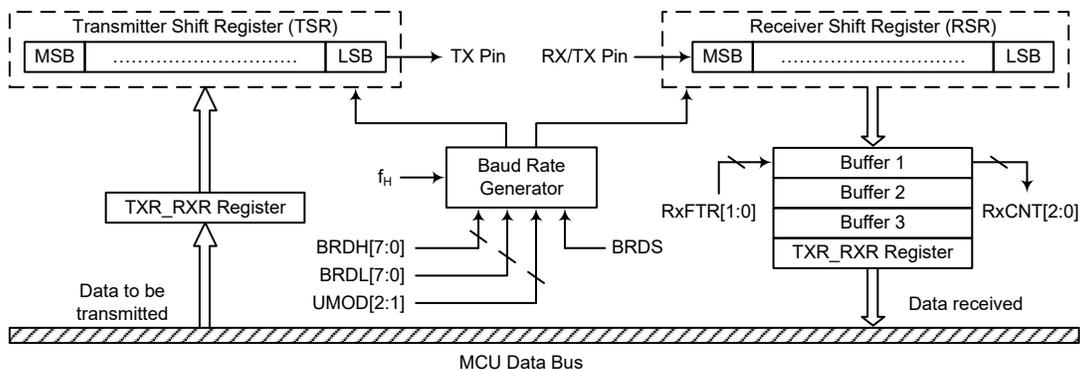
Bit 5~0 **IIC_{TOS5}~IIC_{TOS0}**: I²C time-out period selection
I²C time-out clock source is $f_{SUB}/32$.
I²C time-out time is equal to $(IIC_{TOS}[5:0]+1)\times(32/f_{SUB})$.

UART Interface

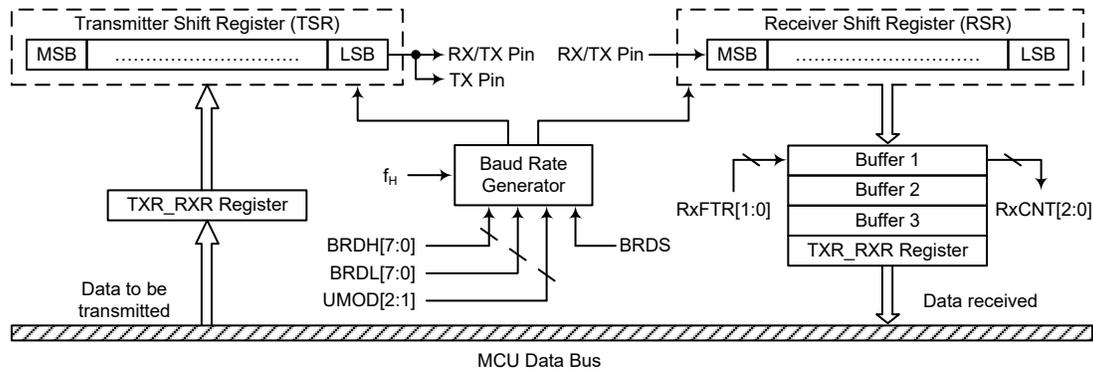
The device contains an integrated full-duplex or half-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data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

The integrated UART function contains the following features:

- Full-duplex or half-duplex (single wire mode), asynchronous communication
- 8 or 9 bits character length
- Even, odd, mark, space or no parity options
- One or two stop bits configurable for receiver
- Two stop bits for transmitter
- Baud rate generator with 16-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 4-byte Deep FIFO Receive Data Buffer
- 1-byte Deep FIFO Transmit Data Buffer
- RX/TX pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
 - ♦ Transmitter Empty
 - ♦ Transmitter Idle
 - ♦ Receiver reaching FIFO trigger level
 - ♦ Receiver Overrun
 - ♦ Address Mode Detect



UART Data Transfer Block Diagram – SWM=0



UART Data Transfer Block Diagram – SWM=1

UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as TX and RX/TX, which are pin-shared with I/O or other pin functions. The TX and RX/TX pin function should first be selected by the corresponding pin-shared function selection register before the UART function is used. Along with the UARTEN bit, the TXEN and RXEN bits, if set, will configure these pins to transmitter output and receiver input conditions. At this time the internal pull-high resistor related to the transmitter output pin will be disabled, while the internal pull-high resistor related to the receiver input pin is controlled by the corresponding I/O pull-high function control bit. When the TX or RX/TX pin function is disabled by clearing the UARTEN, TXEN or RXEN bit, the TX or RX/TX pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the TX or RX/TX pin or not is determined by the corresponding I/O pull-high function control bit.

UART Single Wire Mode

The UART function also supports a Single Wire Mode communication which is selected using the SWM bit in the UCR3 register. When the SWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single RX/TX pin can be used to transmit and receive data depending upon the corresponding control bits. When the RXEN bit is set high, the RX/TX pin is used as a receiver pin. When the RXEN bit is cleared to zero and the TXEN bit is set high, the RX/TX pin will act as a transmitter pin.

It is recommended not to set both the RXEN and TXEN bits high in the single wire mode. If both the RXEN and TXEN bits are set high, the RXEN 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 TX pin mentioned in this chapter should be replaced by the RX/TX pin to understand the whole UART single wire mode function.

In the single wire mode, the data can also be transmitted on the TX pin in a transmission operation with proper software configurations. Therefore, the data will be output on the RX/TX and TX 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 TXR_RXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate

Generator. Only the TXR_RXR register is mapped onto the MCU Data Memory, the Transmit Shift Register is not mapped and is therefore inaccessible to the application program.

Data to be received by the UART is accepted on the external RX/TX pin, from where it is shifted in, LSB first, to the Receiver Shift Register at a rate controlled by the Baud Rate Generator. When the shift register is full, the data will then be transferred from the shift register to the internal TXR_RXR register, where it is buffered and can be manipulated by the application program. Only the TXR_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, TXR_RXR, in the Data Memory.

UART Status and Control Registers

There are nine control registers associated with the UART function. The SWM bit in the UCR3 register is used to enable/disable the UART Single Wire Mode. The USR, UCR1, UCR2, UFCCR and RxCNT registers control the overall function of the UART, while the BRDH and BRDL registers control the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXR data register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
UCR3	—	—	—	—	—	—	—	SWM
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRDH	D7	D6	D5	D4	D3	D2	D1	D0
BRDL	D7	D6	D5	D4	D3	D2	D1	D0
UFCCR	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
RxCNT	—	—	—	—	—	D2	D1	D0

UART Register List

• USR Register

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

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

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

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

- Bit 6 **NF:** Noise flag
 0: No noise is detected
 1: Noise is detected
- The NF flag is the noise flag. When this read only flag is “0”, it indicates no noise condition. When the flag is “1”, it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of an overrun. The NF flag can be cleared by a software sequence which will involve a read to the status register USR followed by an access to the TXR_RXR data register.
- Bit 5 **FERR:** Framing error flag
 0: No framing error is detected
 1: Framing error is detected
- The FERR flag is the framing error flag. When this read only flag is “0”, it indicates that there is no framing error. When the flag is “1”, it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register USR followed by an access to the TXR_RXR data register.
- Bit 4 **OERR:** Overrun error flag
 0: No overrun error is detected
 1: Overrun error is detected
- The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is “0”, it indicates that there is no overrun error. When the flag is “1”, it indicates that an overrun error occurs which will inhibit further transfers to the TXR_RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register USR followed by an access to the TXR_RXR data register.
- Bit 3 **RIDLE:** Receiver status
 0: Data reception is in progress (Data being received)
 1: No data reception is in progress (Receiver is idle)
- The RIDLE flag is the receiver status flag. When this read only flag is “0”, it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is “1”, it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is “1” indicating that the UART receiver is idle and the RX/TX pin stays in logic high condition.
- Bit 2 **RXIF:** Receive TXR_RXR data register status
 0: TXR_RXR data register is empty
 1: TXR_RXR data register has available data and Receiver FIFO trigger level is reached
- The RXIF flag is the receive data register status flag. When this read only flag is “0”, it indicates that the TXR_RXR read data register is empty. When the flag is “1”, it indicates that the TXR_RXR read data register contains new data and Receiver FIFO trigger level is reached. When the contents of the shift register are transferred to the TXR_RXR register and Receiver FIFO trigger level is reached, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag is cleared when the USR register is read with RXIF set, followed by a read from the TXR_RXR register, and if the TXR_RXR register has no data available.
- Bit 1 **TIDLE:** Transmission idle
 0: Data transmission is in progress (Data being transmitted)
 1: No data transmission is in progress (Transmitter is idle)
- The TIDLE flag is known as the transmission complete flag. When this read only flag is “0”, it indicates that a transmission is in progress. This flag will be set high when the TXIF flag is “1” and when there is no transmit data or break character being

transmitted. When TIDLE is equal to “1”, the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared by reading the USR register with TIDLE set and then writing to the TXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 TXIF: Transmit TXR_RXR data register status
 0: Character is not transferred to the transmit shift register
 1: Character has transferred to the transmit shift register (TXR_RXR data register is empty)

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

• **UCR1 Register**

The UCR1 register together with the UCR2 and UCR3 register are the three UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length, 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	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

“x”: unknown

Bit 7 UARTEN: UART function enable control
 0: Disable UART. TX and RX/TX pins are in a floating state
 1: Enable UART. TX and RX/TX pins function as UART pins

The UARTEN bit is the UART enable bit. When this bit is equal to “0”, the UART will be disabled and the RX/TX pin as well as the TX pin will be set in a floating state. When the bit is equal to “1”, the UART will be enabled and the TX and RX/TX pins will function as defined by the SWM mode selection bit together with the TXEN and RXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits as well as the RxCNT register will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6 BNO: Number of data transfer bits selection
 0: 8-bit data transfer
 1: 9-bit data transfer

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

Note that the 9th bit of data if BNO=1, or the 8th bit of data if BNO=0, which is used as the parity bit, does not transfer to RX8 or TXRX7 respectively when the parity function is enabled.

- Bit 5 **PREN**: Parity function enable control
 0: Parity function is disabled
 1: Parity function is enabled
 This is the parity enable bit. When this bit is equal to “1”, the parity function will be enabled. If the bit is equal to “0”, then the parity function will be disabled. Replace the most significant bit position with a parity bit.
- Bit 4~3 **PRT1~PRT0**: Parity type selection bits
 00: Even parity for parity generator
 01: Odd parity for parity generator
 10: Mark parity for parity generator
 11: Space parity for parity generator
 These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.
- Bit 2 **TXBRK**: Transmit break character
 0: No break character is transmitted
 1: Break characters transmit
 The TXBRK bit is the Transmit Break Character bit. When this bit is “0”, there are no break characters and the TX pin operates normally. When the bit is “1”, there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to “1”, after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.
- Bit 1 **RX8**: Receive data bit 8 for 9-bit data transfer format (read only)
 This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.
- Bit 0 **TX8**: Transmit data bit 8 for 9-bit data transfer format (write only)
 This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UCR2 Register**

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

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

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

- Bit 6 **RXEN:** UART Receiver enabled control
 0: UART receiver is disabled
 1: UART receiver is enabled
The bit named RXEN is the Receiver Enable Bit. When this bit is equal to “0”, the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RX/TX pin will be set in a floating state. If the RXEN bit is equal to “1” and the UARTEN bit is also equal to “1”, the receiver will be enabled and the RX/TX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX/TX pin will be set in a floating state.
- Bit 5 **STOPS:** Number of Stop bits selection for receiver
 0: One stop bit format is used
 1: Two stop bits format is used
This bit determines if one or two stop bits are to be used for receiver. When this bit is equal to “1”, two stop bits are used. If this bit is equal to “0”, then only one stop bit is used. Two stop bits are used for transmitter.
- Bit 4 **ADDEN:** Address detect function enable control
 0: Address detect function is disabled
 1: Address detect function is enabled
The bit named ADDEN is the address detect function enable control bit. When this bit is equal to “1”, the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXRX7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of “1”, then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. 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 **WAKE:** RX/TX pin wake-up UART function enable control
 0: RX/TX pin wake-up UART function is disabled
 1: RX/TX pin wake-up UART function is enabled
This bit is used to control the wake-up UART function when a falling edge on the RX/TX pin occurs. Note that this bit is only available when the UART clock (f_{H}) is switched off. There will be no RX/TX pin wake-up UART function if the UART clock (f_{H}) exists. If the WAKE bit is set to 1 as the UART clock (f_{H}) is switched off, a UART wake-up request will be initiated when a falling edge on the RX/TX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX/TX 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 RX/TX pin when the WAKE bit is cleared to 0.
- Bit 2 **RIE:** Receiver interrupt enable control
 0: Receiver related interrupt is disabled
 1: Receiver related interrupt is enabled
This bit enables or disables the receiver interrupt. If this bit is equal to “1” and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.
- Bit 1 **TIE:** 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 TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

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

• **UCR3 Register**

The UCR3 register is used to enable the UART Single Wire Mode communication. As the name suggests in the single wire mode the UART communication can be implemented in one single line, RX/TX, together with the control of the RXEN and TXEN bits in the UCR2 register.

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

Bit 7~1 Unimplemented, read as “0”

Bit 0 **SWM**: Single Wire Mode enable control
 0: Disable, the RX/TX pin is used as UART receiver function only
 1: Enable, the RX/TX pin can be used as UART receiver or transmitter function controlled by the RXEN and TXEN bits

Note that when the Single Wire Mode is enabled, if both the RXEN and TXEN bits are high, the RX/TX pin will just be used as UART receiver input.

• **TXR_RXR Register**

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

Bit	7	6	5	4	3	2	1	0
Name	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
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 **TXRX7~TXRX0**: UART Transmit/Receive Data bit 7 ~ bit 0

• **BRDH Register**

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

Bit 7~0 **D7~D0**: Baud rate divider high byte

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

$$\text{Baud Rate} = f_{\text{clk}} / (\text{BRD} + \text{UMOD} / 8)$$

BRD=16~65535 or 8~65535 depending on BRDS

Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.

2. The BRDL must be written first and then BRDH, otherwise errors may occur.

3. The BRDH register should not be modified during data transmission process.

• **BRDL Register**

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

Bit 7~0 **D7~D0**: Baud rate divider low byte
 The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.
 $Baud\ Rate = f_{IH} / (BRD + UMOD/8)$
 $BRD = 16 \sim 65535$ or $8 \sim 65535$ depending on BRDS
 Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.
 2. The BRDL must be written first and then BRDH, otherwise errors may occur.
 3. The BRDL register should not be modified during data transmission process.

• **UFCR Register**

The UFCR register is the FIFO control register which is used for UART modulation control, BRD range selection and trigger level selection for RXIF and interrupt.

Bit	7	6	5	4	3	2	1	0
Name	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as “0”
 Bit 5~3 **UMOD2~UMOD0**: UART Modulation Control bits
 The modulation control bits are used to correct the baud rate of the received or transmitted UART signal. These bits determine if the extra UART clock cycle should be added in a UART bit time. The UMOD2~UMOD0 will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle.
 Bit 2 **BRDS**: BRD range selection
 0: BRD range is from 16 to 65535
 1: BRD range is from 8 to 65535
 The BRDS is used to control the sampling point in a UART bit time. If the BRDS bit is cleared to zero, the sampling point will be $BRD/2$, $BRD/2+1 \times f_{IH}$, and $BRD/2+2 \times f_{IH}$ in a UART bit time. If the BRDS bit is set high, the sampling point will be $BRD/2-1 \times f_{IH}$, $BRD/2$, and $BRD/2+2 \times f_{IH}$ in a UART bit time.
 Note that the BRDS bit should not be modified during data transmission process.
 Bit 1~0 **RxFTR1~RxFTR0**: Receiver FIFO trigger level (bytes)
 00: 4 bytes in Receiver FIFO
 01: 1 or more bytes in Receiver FIFO
 10: 2 or more bytes in Receiver FIFO
 11: 3 or more bytes in Receiver FIFO
 For the receiver these bits define the number of received data bytes in the Receiver FIFO that will trigger the RXIF bit being set high, an interrupt will also be generated if the RIE bit is enabled. To prevent OERR from being set high, the receiver FIFO trigger level can be set to 2 bytes, avoiding an overrun state that cannot be processed by the program in time when more than 4 data bytes are received. After the reset the Receiver FIFO is empty.

• **RxCNT Register**

The RxCNT register is the counter used to indicate the number of received data bytes in the Receiver FIFO which have not been read by the MCU. This register is read only.

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

Bit 7~3 Unimplemented, read as “0”

Bit 2~0 **D2~D0**: Receiver FIFO counter

The RxCNT register is the counter used to indicate the number of received data bytes in the Receiver FIFO which is not read by the MCU. When Receiver FIFO receives one byte data, the RxCNT will increase by one; when the MCU reads one byte data from the Receiver FIFO, the RxCNT will decrease by one. If there are 4 bytes of data in the Receiver FIFO, the 5th data will be saved in the shift register. If there is 6th data, the 6th data will be saved in the shift register. But the RxCNT remains the value of 4. The RxCNT will be cleared when reset occurs or UARTEN=1. This register is read only.

Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRDH/BRDL register and the second is the UART modulation control bits UMOD2~UMOD0. To prevent accumulated error of the receiver baud rate frequency, it is recommended to use two stop bits for resynchronization after each byte is received. If a baud rate BR is required with UART clock f_H .

$$f_H/BR = \text{Integer Part} + \text{Fractional Part}$$

The integer part is loaded into BRD (BRDH/BRDL). The fractional part is multiplied by 8 and rounded, then loaded into the UMOD bit field below:

$$BRD = \text{TRUNC}(f_H/BR)$$

$$UMOD = \text{ROUND}[\text{MOD}(f_H/BR) \times 8]$$

Therefore, the actual baud rate is calculated as follows:

$$\text{Baud rate} = f_H / [BRD + (UMOD/8)]$$

Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, determine the BRDH/BRDL register value, the actual baud rate and the error value for a desired baud rate of 230400.

From the above formula, the $BRD = \text{TRUNC}(f_H/BR) = \text{TRUNC}(17.36111) = 17$

The $UMOD = \text{ROUND}[\text{MOD}(f_H/BR) \times 8] = \text{ROUND}(0.36111 \times 8) = \text{ROUND}(2.88888) = 3$

The actual Baud Rate $= f_H / [BRD + (UMOD/8)] = 230215.83$

Therefore the error is equal to $(230215.83 - 230400) / 230400 = -0.08\%$

Modulation Control Example

To get the best-fitting bit sequence for UART modulation control bits UMOD2~UMOD0, the following algorithm can be used: Firstly, the fractional part of the theoretical division factor is multiplied by 8. Then the product will be rounded and UMOD2~UMOD0 bits will be filled with the rounded value. The UMOD2~UMOD0 will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle. The following is an example using the fraction 0.36111 previously calculated: $UMOD[2:0] = \text{ROUND}(0.36111 \times 8) = 011b$.

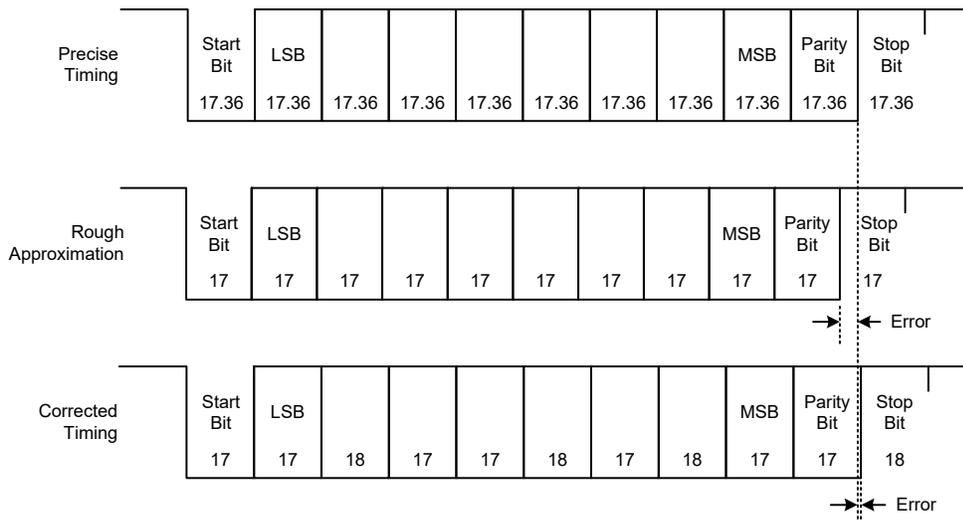
Fraction Addition	Carry to Bit 3	UART Bit Time Sequence	Extra UART Clock Cycle
0000b+0011b=0011b	No	Start bit	No
0011b+0011b=0110b	No	D0	No
0110b+0011b=1001b	Yes	D1	Yes
1001b+0011b=1100b	No	D2	No
1100b+0011b=1111b	No	D3	No
1111b+0011b=0010b	Yes	D4	Yes
0010b+0011b=0101b	No	D5	No
0101b+0011b=1000b	Yes	D6	Yes
1000b+0011b=1011b	No	D7	No
1011b+0011b=1110b	No	Parity bit	No
1110b+0011b=0001b	Yes	Stop bit	Yes

Baud Rate Correction Example

The following figure presents an example using a baud rate of 230400 generated with UART clock f_H . The data format for the following figure is: eight data bits, parity enabled, no address bit, two stop bits.

The following figure shows three different frames:

- The upper frame is the correct one, with a bit-length of 17.36 f_H cycles ($4000000/230400=17.36$).
- The middle frame uses a rough estimate, with 17 f_H cycles for the bit length.
- The lower frame shows a corrected frame using the best fit for the UART modulation control bits UMOD2~UMOD0.



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, mark, space or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits along with the parity are setup by programming the BNO, PRT1~PRT0 and PREN bits. The transmitter always uses two stop bits while the receiver uses one or two stop bits which is determined by the STOPS bit. The baud rate used to transmit and receive data is setup using the internal 16-bit baud rate

generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

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

Clearing the UARTEN bit will disable the TX and RX/TX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF as well as register RxCNT being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

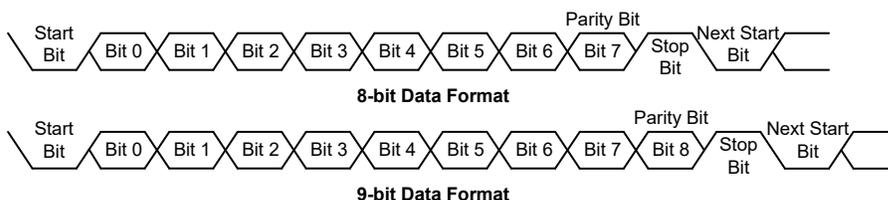
Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1 and UCR2 registers. The BNO bit controls the number of data bits which can be set to either 8 or 9, the PRT1~PRT0 bits control the choice of odd, even, mark or space parity, the PREN bit controls the parity on/off function and the STOPS bit decides whether one or two stop bits are to be used for the receiver, while the transmitter always uses two stop bits. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only configurable for the receiver. The transmitter uses two stop bits.

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1 or 2
1	7	0	1	1 or 2
1	7	1	0	1 or 2
Example of 9-bit Data Formats				
1	9	0	0	1 or 2
1	8	0	1	1 or 2
1	8	1	0	1 or 2

Transmitter Receiver Data Format

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



UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR_RXR register. The data to be transmitted is loaded into this TXR_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 TXR_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 TXEN bit is set, but the data will not be transmitted until the TXR_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 TXR_RXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared functions by configuring the corresponding pin-shared control bits.

Transmitting Data

When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant bit first. In the transmit mode, the TXR_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 TX8 bit in the UCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT1~PRT0 and PREN bits to define the required word length and parity type. Two stop bits are used for the transmitter.
- Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.
- Set the TXEN bit to ensure that the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR_RXR register. Note that this step will clear the TXIF bit.

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

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

1. A USR register access
2. A TXR_RXR register write execution

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

During a data transmission, a write instruction to the TXR_RXR register will place the data into the TXR_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 TXR_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission,

and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLE bit the following software sequence is used:

1. A USR register access
2. A TXR_RXR register write execution

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

Transmitting Break

If the TXBRK bit is set and the state keeps for a time greater than $(BRD+1) \times t_{TH}$ while TIDLE=1, then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by $13 \times N$ '0' bits and stop bits, where $N=1, 2, \text{etc.}$ If a break character is to be transmitted then the TXBRK bit must be first set by the application program, and then cleared to generate the stop bits. Transmitting a break character will not generate a transmit interrupt. Note that a break condition length is at least 13 bits long. If the TXBRK bit is continually kept at a logic high level then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the TXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out 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 BNO bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8 bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX/TX external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX/TX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX/TX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX/TX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

Receiving Data

When the UART receiver is receiving data, the data is serially shifted in on the external RX/TX input pin, LSB first. In the read mode, the TXR_RXR register forms a buffer between the internal bus and the receiver shift register. The TXR_RXR register is a four byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR_RXR before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERR will be subsequently indicated. For continuous multi-byte data transmission, it is strongly recommended that the receiver uses two stop bits to avoid a receiving error caused by the accumulated error of the receiver baud rate frequency.

The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNO, PRT1~PRT0, PREN and STOPS bits to define the word length and parity type and number of stop bits.
- Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.
- Set the RXEN bit to ensure that the RX/TX pin is used as a UART receiver pin.

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

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

- The RXIF bit in the USR register will be set when the TXR_RXR register has data available, the number of the available data bytes can be checked by polling the RxCNT register content.
- When the contents of the shift register have been transferred to the TXR_RXR register and reach Receiver FIFO trigger level if the RIE bit is set, then an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

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

1. A USR register access
2. A TXR_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 BNO plus one or two stop bits. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO plus one or two stop bits. The RXIF bit is set, FERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLE bit is set. A break is regarded as a character that contains only zeros with the FERR flag set. If a long break signal has been detected, the receiver will regard it as a data frame including a start bit, data bits and the invalid stop bit and the FERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until one or two stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

- The framing error flag, FERR, will be set.
- The receive data register, TXR_RXR, will be cleared.
- The OERR, NF, PERR, RIDLE or RXIF flags will possibly be set.

Idle Status

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

Receiver Interrupt

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

When a subroutine will be called with an execution time longer than the time for UART to receive five data bytes, if the UART received data could not be read in time during the subroutine execution, clear the RXEN bit to zero in advance to suspend data reception. If the UART interrupt could not be served in time to process the overrun error during the subroutine execution, ensure that both EMI and RXEN bits are disabled during this period, and then enable EMI and RXEN again after the subroutine execution has been completed to continue the UART data reception.

Managing Receiver Errors

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

Overrun Error – OERR

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

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

- The OERR flag in the USR register will be set.
- The TXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIE bit is set.

When the OERR flag is set to “1”, it is necessary to read five data bytes from the four-byte deep receiver FIFO and the shift register immediately to avoid unexpected errors, such as the UART is unable to receive data. If such an error occurs, clear the RXEN bit to “0” then set it to “1” again to continue data reception.

The OERR flag can be cleared by an access to the USR register followed by a read to the TXR_RXR register.

Noise Error – NF

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

- The read only noise flag, NF, in the USR register will be set on the rising edge of the RXIF bit.
- Data will be transferred from the Shift register to the TXR_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the RXIF bit which itself generates an interrupt.

Note that the NF flag is reset by a USR register read operation followed by a TXR_RXR register read operation.

Framing Error – FERR

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

Parity Error – PERR

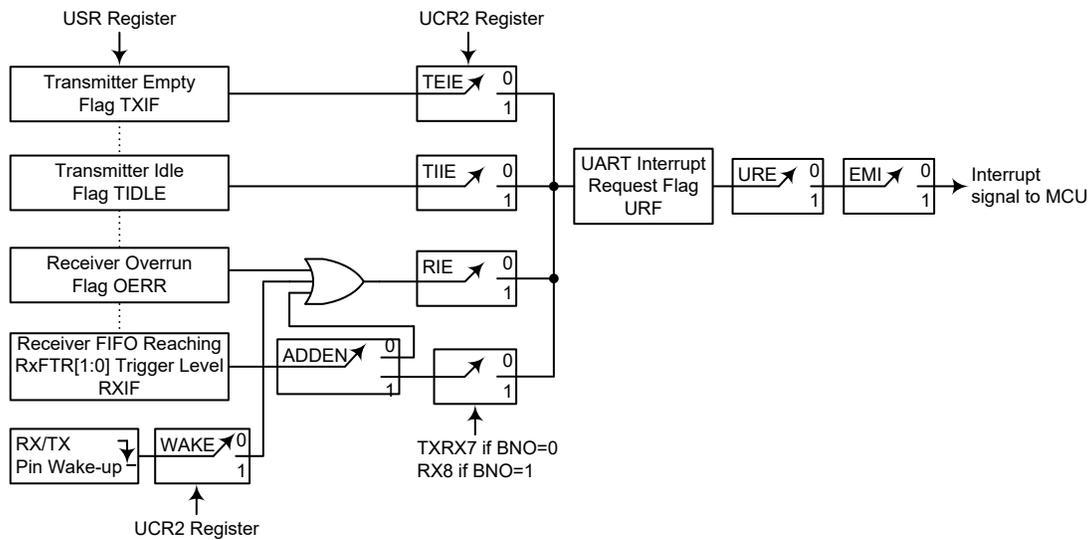
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd, even, mark or space, is selected. The read only PERR flag and the received data will be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

UART Interrupt Structure

Several individual UART conditions can generate a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and its corresponding interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding USR register flags which will generate a UART interrupt if its associated interrupt enable control bit in the UCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual UART interrupt sources.

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX/TX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the UART clock (f_{IH}) source is switched off and the WAKE and RIE bits in the UCR2 register are set when a falling edge on the RX/TX pin occurs.

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



UART Interrupt Structure

Address Detect Mode

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the URE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to zero.

ADDEN	9th Bit if BNO=1 8th Bit if BNO=0	UART Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

ADDEN Bit Function

UART Power Down and Wake-up

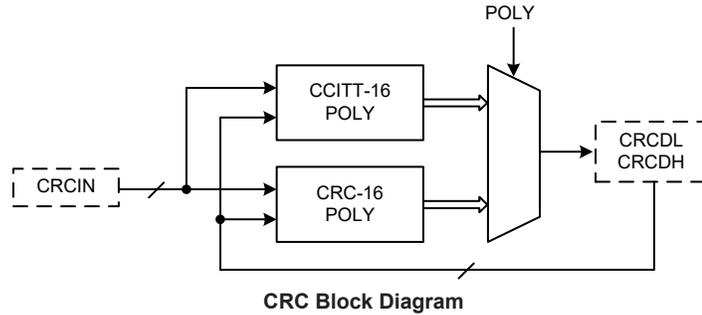
When the UART clock (f_{H}) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock (f_{H}) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. 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 USR, UCR1, UCR2, UCR3, UFCR, RxCNT, TXR_RXR as well as the BRDH and BRDL registers will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX/TX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the UART clock (f_{H}) is off, then a falling edge on the RX/TX pin will trigger an RX/TX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX/TX pin will be ignored.

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

Cyclic Redundancy Check – CRC

The Cyclic Redundancy Check, CRC, calculation unit is an error detection technique test algorithm and uses to verify data transmission or storage data correctness. A CRC calculation takes a data stream or a block of data as input and generates a 16-bit output remainder. Ordinarily, a data stream is suffixed by a CRC code and used as a checksum when being sent or stored. Therefore, the received or restored data stream is calculated by the same generator polynomial as described in the following section.



CRC Registers

The CRC generator contains an 8-bit CRC data input register, CRCIN, and a CRC checksum register pair, CRCDH and CRCDL. The CRCIN register is used to input new data and the CRCDH and CRCDL registers are used to hold the previous CRC calculation result. A CRC control register, CRCCR, is used to select which CRC generating polynomial is used.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CRCCR	—	—	—	—	—	—	—	POLY
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D15	D14	D13	D12	D11	D10	D9	D8

CRC Register List

• CRCCR Register

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

Bit 7~1 Unimplemented, read as “0”

Bit 0 **POLY**: 16-bit CRC generating polynomial selection
 0: CRC-CCITT: $X^{16}+X^{12}+X^5+1$
 1: CRC-16: $X^{16}+X^{15}+X^2+1$

• CRCIN Register

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

Bit 7~0 **D7~D0**: CRC input data register

• **CRCDL Register**

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

Bit 7~0 **D7~D0**: 16-bit CRC checksum low byte data register

• **CRCDH Register**

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

Bit 7~0 **D15~D8**: 16-bit CRC checksum high byte data register

CRC Operation

The CRC generator provides the 16-bit CRC result calculation based on the CRC16 and CCITT CRC16 polynomials. In this CRC generator, there are only these two polynomials available for the numeric values calculation. It cannot support the 16-bit CRC calculations based on any other polynomials.

The following two expressions can be used for the CRC generating polynomial which is determined using the POLY bit in the CRC control register, CRCCR. The CRC calculation result is called as the CRC checksum, CRCSUM, and stored in the CRC checksum register pair, CRCDH and CRCDL.

- CRC-CCITT: $X^{16}+X^{12}+X^5+1$.
- CRC-16: $X^{16}+X^{15}+X^2+1$.

CRC Computation

Each write operation to the CRCIN register creates a combination of the previous CRC value stored in the CRCDH and CRCDL registers and the new data input. The CRC unit calculates the CRC data register value is based on byte by byte. It will take one MCU instruction cycle to calculate the CRC checksum.

CRC Calculation Procedures:

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Execute an “Exclusive OR” operation with the 8-bit input data byte and the 16-bit CRCSUM high byte. The result is called the temporary CRCSUM.
3. Shift the temporary CRCSUM value left by one bit and move a “0” into the LSB.
4. Check the shifted temporary CRCSUM value after procedure 3.

If the MSB is 0, then this shifted temporary CRCSUM will be considered as a new temporary CRCSUM.

Otherwise, execute an “Exclusive OR” operation with the shifted temporary CRCSUM in procedure 3 and a data “8005H”. Then the operation result will be regarded as the new temporary CRCSUM.

Note that the data to be perform an “Exclusive OR” operation is “8005H” for the CRC-16 polynomial while for the CRC-CCITT polynomial the data is “1021H”.

5. Repeat the procedure 3 ~ procedure 4 until all bits of the input data byte are completely calculated.

- Repeat the procedure 2 ~ procedure 5 until all of the input data bytes are completely calculated. Then, the latest calculated result is the final CRC checksum, CRCSUM.

CRC Calculation Examples:

- Write 1 byte input data into the CRCIN register and the corresponding CRC checksum are individually calculated as the following table shown.

CRC Data Input CRC Polynomial	00H	01H	02H	03H	04H	05H	06H	07H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	0000H	1021H	2042H	3063H	4084H	50A5H	60C6H	70E7H
CRC-16 ($X^{16}+X^{15}+X^2+1$)	0000H	8005H	800FH	000AH	801BH	001EH	0014H	8011H

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before each CRC input data is written into the CRCIN register.

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

CRC Data Input CRC Polynomial	CRCIN=78h→56h→34h→12h
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	(CRCDH, CRCDL)=FF9FH→BBC3H→A367H→D0FAH
CRC-16 ($X^{16}+X^{15}+X^2+1$)	(CRCDH, CRCDL)=0110h→91F1h→F2DEh→5C43h

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before the sequential CRC data input operation.

Program Memory CRC Checksum Calculation Example:

- Clear the checksum register pair, CRCDH and CRCDL.
- Select the CRC-CCITT or CRC-16 polynomial as the generating polynomial using the POLY bit in the CRCCR register.
- Execute the table read instruction to read the program memory data value.
- Write the table data low byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.
- Write the table data high byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.
- Repeat the procedure 3 ~ procedure 5 to read the next program memory data value and execute the CRC calculation until all program memory data are read followed by the sequential CRC calculation. Then the value in the CRC checksum register pair is the final CRC calculation result.

Low Voltage Detector – LVD

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

LVD Register

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

• LVDC Register

Bit	7	6	5	4	3	2	1	0
Name	TLVD1	TLVD0	LVDO	LVDEN	—	VLVD2	VLVD1	VLVD0
R/W	R/W	R/W	R	R/W	—	R/W	R/W	R/W
POR	0	0	0	0	—	0	0	0

Bit 7~6 **TLVD1~TLVD0**: Select Minimum low voltage width to interrupt (t_{LVD})

00: $(1\sim2) \times t_{LIRC}$

01: $(3\sim4) \times t_{LIRC}$

10: $(7\sim8) \times t_{LIRC}$

11: $(1\sim2) \times t_{LIRC}$

Bit 5 **LVDO**: LVD output flag

0: No Low Voltage Detected

1: Low Voltage Detected

Bit 4 **LVDEN**: Low voltage detector enable control

0: Disable

1: Enable

Bit 3 Unimplemented, read as “0”

Bit 2~0 **VLVD2~VLVD0**: LVD voltage selection

000: 1.8V

001: 2.0V

010: 2.4V

011: 2.7V

100: 3.0V

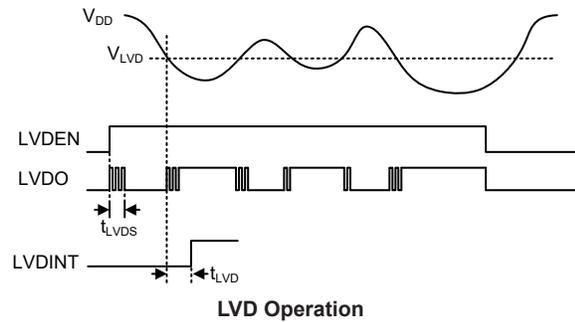
101: 3.3V

110: 3.6V

111: 4.0V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD} , with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.8V and 4.0V. When the power supply voltage, V_{DD} , falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. When the device enters the SLEEP mode, the low voltage detector will be automatically disabled even if the LVDEN bit is set high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} may rise and fall rather slowly, at the voltage nears that of V_{LVD} , there may be multiple bit LVDO transitions.

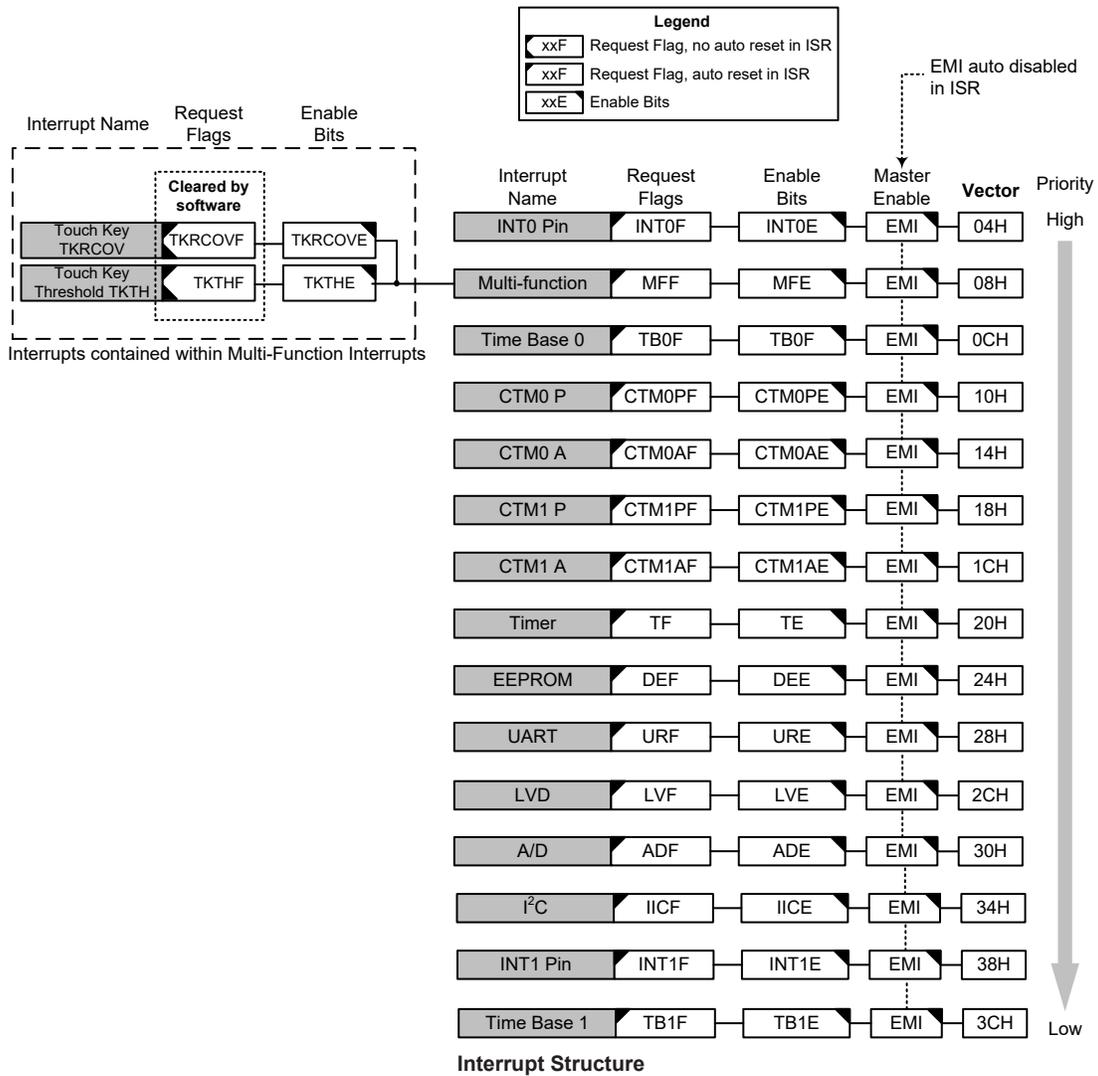


The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} falls below the preset LVD voltage. This will cause the device to wake up from the IDLE Mode, however if the Low Voltage Detector wake-up function is not required then the LVF flag should be first set high before the device enters the IDLE Mode.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the Touch Keys, TMs, Time Bases, LVD, EEPROM, UART and the A/D converter, etc.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector.



Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The registers fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI register which setup the Multi-function interrupt. Finally there is an INTEG register to setup the external interrupt trigger edge type.

Each register contains a number of enable bits to enable or disable individual registers as well as interrupt flags to indicate the presence of an interrupt request. The naming convention of these follows a specific pattern. First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an “E” for enable/disable bit or “F” for request flag.

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pin	INTnE	INTnF	n=0~1
A/D Converter	ADE	ADF	—
Multi-function	MFE	MFF	—
Time Base	TBnE	TBnF	n=0~1
UART	URE	URF	—
LVD	LVE	LVF	—
EEPROM erase or write operation	DEE	DEF	—
CTM	CTMnAE	CTMnAF	n=0~1
	CTMnPE	CTMnPF	
Timer	TE	TF	—
I ² C Interface	IICE	IICF	—
Touch key TKRCOV	TKRCOVE	TKRCOVF	—
Touch key threshold TKTH	TKTHE	TKTHF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	TB0F	MFF	INT0F	TB0E	MFE	INT0E	EMI
INTC1	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
INTC2	LVF	URF	DEF	TF	LVE	URE	DEE	TE
INTC3	TB1F	INT1F	IICF	ADF	TB1E	INT1E	IICE	ADE
MFI	—	—	TKTHF	TKRCOVF	—	—	TKTHE	TKRCOVE

Interrupt Register List

• INTEG Register

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

Bit 7~4 Unimplemented, read as “0”

Bit 3~2 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
 00: Disable
 01: Rising edge
 10: Falling edge
 11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt edge control for INT0 pin
 00: Disable
 01: Rising edge
 10: Falling edge
 11: Rising and falling edges

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TB0F	MFF	INT0F	TB0E	MFE	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 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request
- Bit 5 **MFF**: Multi-function interrupt request flag
0: No request
1: Interrupt request
- Bit 4 **INT0F**: INT0 interrupt request flag
0: No request
1: Interrupt request
- Bit 3 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable
- Bit 2 **MFE**: Multi-function 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	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
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 **CTM1AF**: CTM1 comparator A match interrupt request flag
0: No request
1: Interrupt request
- Bit 6 **CTM1PF**: CTM1 comparator P match interrupt request flag
0: No request
1: Interrupt request
- Bit 5 **CTM0AF**: CTM0 comparator A match interrupt request flag
0: No request
1: Interrupt request
- Bit 4 **CTM0PF**: CTM0 comparator P match interrupt request flag
0: No request
1: Interrupt request
- Bit 3 **CTM1AE**: CTM1 comparator A match interrupt control
0: Disable
1: Enable
- Bit 2 **CTM1PE**: CTM1 comparator P match interrupt control
0: Disable
1: Enable

- Bit 1 **CTM0AE**: CTM0 comparator A match interrupt control
 0: Disable
 1: Enable
- Bit 0 **CTM0PE**: CTM0 comparator P match interrupt control
 0: Disable
 1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	LVF	URF	DEF	TF	LVE	URE	DEE	TE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7 **LVF**: LVD interrupt request flag
 0: No request
 1: Interrupt request
- Bit 6 **URF**: UART interrupt request flag
 0: No request
 1: Interrupt request
- Bit 5 **DEF**: Data EEPROM interrupt request flag
 0: No request
 1: Interrupt request
- Bit 4 **TF**: Timer interrupt request Flag
 0: No request
 1: Interrupt request
- Bit 3 **LVE**: LVD interrupt Control
 0: Disable
 1: Enable
- Bit 2 **URE**: UART interrupt control
 0: Disable
 1: Enable
- Bit 1 **DEE**: Data EEPROM interrupt control
 0: Disable
 1: Enable
- Bit 0 **TE**: Timer interrupt Control
 0: Disable
 1: Enable

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	TB1F	INT1F	IICF	ADF	TB1E	INT1E	IICE	ADE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7 **TB1F**: Time Base 1 interrupt request flag
 0: No request
 1: Interrupt request
- Bit 6 **INT1F**: INT1 interrupt request flag
 0: No request
 1: Interrupt request
- Bit 5 **IICF**: I²C interrupt request flag
 0: No request
 1: Interrupt request

- Bit 4 **ADF**: A/D converter interrupt request flag
0: No request
1: Interrupt request
- Bit 3 **TBIE**: Time Base 1 interrupt control
0: Disable
1: Enable
- Bit 2 **INTIE**: INT1 interrupt control
0: Disable
1: Enable
- Bit 1 **IICE**: I²C interrupt control
0: Disable
1: Enable
- Bit 0 **ADE**: A/D converter interrupt control
0: Disable
1: Enable

• **MFI Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	TKTHF	TKRCOVF	—	—	TKTHE	TKRCOVE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6 Unimplemented, read as “0”
- Bit 5 **TKTHF**: Touch key threshold TKTH interrupt request flag
0: No request
1: Interrupt request

Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 4 **TKRCOVF**: Touch key TKRCOV interrupt request flag
0: No request
1: Interrupt request

Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 3~2 Unimplemented, read as “0”
- Bit 1 **TKTHE**: Touch key threshold TKTH interrupt control
0: Disable
1: Enable
- Bit 0 **TKRCOVE**: Touch key TKRCOV interrupt control
0: Disable
1: Enable

Interrupt Operation

When the conditions for an interrupt event occur, such as a Touch Key Counter overflow or TM Comparator P or Comparator A match or A/D conversion completion etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will

then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a “JMP” which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a “RETI”, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the Interrupt Structure diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.

External Interrupts

The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to their respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bits, INT0E~INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pins, a subroutine call to the corresponding external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. 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 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 Interrupt

Within the device there is one Multi-function interrupt. Unlike the other independent interrupts, this interrupt has no independent source, but rather are formed from other existing interrupt sources, namely the touch key TKRCOV interrupt and touch key threshold TKTH interrupt.

A Multi-function interrupt request will take place when the Multi-function interrupt request flag MFF is set. The Multi-function interrupt flag will be set when any of its included functions generate an interrupt request flag. To allow the program to branch to its interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and one of the interrupts contained within the Multi-function interrupt occurs, a subroutine call to the Multi-function interrupt vector will take place. When the interrupt is serviced, the 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 interrupt will not be automatically reset and must be manually reset by the application program.

Touch Key TKRCOV Interrupt

The Touch Key TKRCOV Interrupt is contained within the Multi-function Interrupt. A Touch Key TKRCOV Interrupt request will take place when the Touch Key TKRCOV Interrupt request flag, TKRCOVF, is set, which occurs when the touch key time slot counter overflows or all the scan operations are completed. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, Touch Key TKRCOV Interrupt enable bit, TKRCOVE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and the touch key time slot counter overflows, a subroutine calls to the Multi-function Interrupt vector, will take place. When the Touch Key TKRCOV Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the TKRCOVF flag will not be automatically cleared, it has to be cleared by the application program.

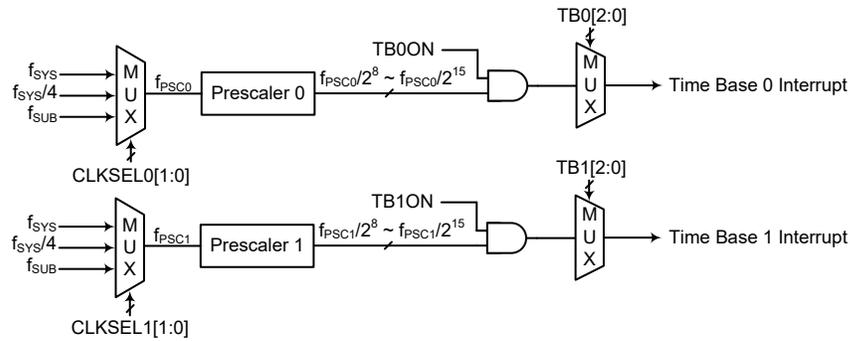
Touch Key Threshold TKTH Interrupt

The Touch Key Threshold TKTH Interrupt is contained within the Multi-function Interrupt. A Touch Key Threshold TKTH Interrupt request will take place when the Touch Key Threshold TKTH Interrupt request flag, TKTHF, is set, which occurs when the Touch Key Module 16-bit C/F counter is less than the lower threshold if M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, Touch Key Threshold TKTH Interrupt enable bit, TKTHE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and any of the above described threshold comparison conditions occurs, a subroutine calls to the Multi-function Interrupt vector, will take place. When the Touch Key Threshold TKTH Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the TKTHF flag will not be automatically cleared, it has to be cleared by the application program.

Time Base Interrupts

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

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, f_{PSC0} or f_{PSC1} , originates from the internal clock source f_{SYS} , $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source that generates f_{PSC0} or f_{PSC1} , which in turn controls the Time Base interrupt period, is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the PSC0R and PSC1R register respectively.



Time Base Interrupts

• PSCnR Register (n=0~1)

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSELn1	CLKSELn0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSELn1~CLKSELn0**: Prescaler n clock source selection

00: f_{SYS}

01: $f_{SYS}/4$

1x: f_{SUB}

• TB0C Register

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

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

0: Disable

1: Enable

Bit 6~3 Unimplemented, read as “0”

Bit 2~0 **TB02~TB00**: Select Time Base 0 Time-out Period
 000: $2^8/f_{PSC0}$
 001: $2^9/f_{PSC0}$
 010: $2^{10}/f_{PSC0}$
 011: $2^{11}/f_{PSC0}$
 100: $2^{12}/f_{PSC0}$
 101: $2^{13}/f_{PSC0}$
 110: $2^{14}/f_{PSC0}$
 111: $2^{15}/f_{PSC0}$

• **TB1C Register**

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

Bit 7 **TB1ON**: Time Base 1 Control
 0: Disable
 1: Enable

Bit 6~3 Unimplemented, read as “0”

Bit 2~0 **TB12~TB10**: Select Time Base 1 Time-out Period
 000: $2^8/f_{PSC1}$
 001: $2^9/f_{PSC1}$
 010: $2^{10}/f_{PSC1}$
 011: $2^{11}/f_{PSC1}$
 100: $2^{12}/f_{PSC1}$
 101: $2^{13}/f_{PSC1}$
 110: $2^{14}/f_{PSC1}$
 111: $2^{15}/f_{PSC1}$

UART Interrupt

The UART Interrupt is controlled by several UART transfer conditions. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and UART Interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of the conditions described above occurs, a subroutine call to the corresponding UART Interrupt vector, will take place. When the interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

I²C Interrupt

An I²C interrupt request will take place when the I²C Interrupt request flag, IICF, is set, which occurs when a byte of data has been received or transmitted by the I²C interface, or an I²C slave address match occurs, or an I²C bus time-out occurs. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the I²C Interrupt enable bit, IICE, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the I²C Interrupt vector, will take place. When the interrupt is serviced, the I²C Interrupt flag, IICF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Timer Interrupt

An actual Timer interrupt will take place when the Timer request flag, TF, is set, which occurs when the Timer overflows. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Timer Interrupt enable bit, TE, must first be set. When the interrupt is enabled, the stack is not full and a Timer overflow occurs, a subroutine call to the relevant timer interrupt vector, will take place. When the interrupt is serviced, the timer interrupt request flag, TF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

LVD Interrupt

An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and Low Voltage Interrupt enable bit, LVE, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the LVD Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the LVD interrupt request flag LVF flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

EEPROM 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 interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, 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 EEPROM Interrupt vector will take place. When the Interrupt is serviced, the EEPROM interrupt request flag DEF will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Timer Module Interrupts

Each Compact Type TM has two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. Each Compact Type TM has two interrupt request flags, CTMnAF and CTMnPF, and two enable control bits, CTMnAE and CTMnPE. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, and the respective TM Interrupt enable bit, CTMnAE or CTMnPE, 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 corresponding TM Interrupt vector location, will take place. When the TM interrupt is serviced, the TM interrupt request flag, CTMnAF or CTMnPF, will be automatically cleared and the EMI bit will be also 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.

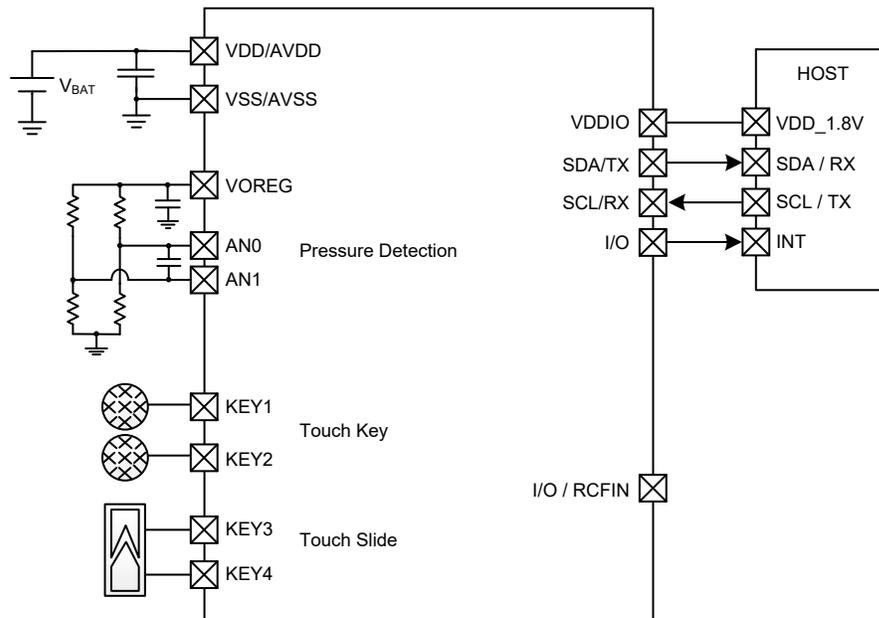
It is recommended that programs do not use the “CALL” instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged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

Application Circuits



Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5 μ s and branch or call instructions would be implemented within 1 μ s. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be “CLR PCL” or “MOV PCL, A”. For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of 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	↑Note	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	↑Note	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	↑Note	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	↑Note	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	↑Note	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	↑Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	↑Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	↑Note	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	↑Note	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	↑Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	↑Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	↑Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	↑Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	↑Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	↑Note	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	1 ^{Note}	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1 ^{Note}	None
SET [m].i	Set bit of Data Memory	1 ^{Note}	None
Branch Operation			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1 ^{Note}	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1 ^{Note}	None
SZ [m].i	Skip if bit i of Data Memory is zero	1 ^{Note}	None
SNZ [m]	Skip if Data Memory is not zero	1 ^{Note}	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1 ^{Note}	None
SIZ [m]	Skip if increment Data Memory is zero	1 ^{Note}	None
SDZ [m]	Skip if decrement Data Memory is zero	1 ^{Note}	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1 ^{Note}	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1 ^{Note}	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	2 ^{Note}	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2 ^{Note}	None
ITABRD [m]	Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory	2 ^{Note}	None
ITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	2 ^{Note}	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1 ^{Note}	None
SET [m]	Set Data Memory	1 ^{Note}	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1 ^{Note}	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	2 ^{Note}	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	2 ^{Note}	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	2 ^{Note}	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	2 ^{Note}	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2 ^{Note}	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	2 ^{Note}	Z
LORM A,[m]	Logical OR ACC to Data Memory	2 ^{Note}	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2 ^{Note}	Z
LCPL [m]	Complement Data Memory	2 ^{Note}	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	2 ^{Note}	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2 ^{Note}	Z
Rotate			
LRRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2 ^{Note}	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2 ^{Note}	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2 ^{Note}	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2 ^{Note}	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2 ^{Note}	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2 ^{Note}	None
LSET [m].i	Set bit of Data Memory	2 ^{Note}	None

Mnemonic	Description	Cycles	Flag Affected
Branch			
LSZ [m]	Skip if Data Memory is zero	2 ^{Note}	None
LSZA [m]	Skip if Data Memory is zero with data movement to ACC	2 ^{Note}	None
LSNZ [m]	Skip if Data Memory is not zero	2 ^{Note}	None
LSZ [m].i	Skip if bit i of Data Memory is zero	2 ^{Note}	None
LSNZ [m].i	Skip if bit i of Data Memory is not zero	2 ^{Note}	None
LSIZ [m]	Skip if increment Data Memory is zero	2 ^{Note}	None
LSDZ [m]	Skip if decrement Data Memory is zero	2 ^{Note}	None
LSIZA [m]	Skip if increment Data Memory is zero with result in ACC	2 ^{Note}	None
LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC	2 ^{Note}	None
Table Read			
LTABRD [m]	Read table (specific page) to TBLH and Data Memory	3 ^{Note}	None
LTABRDL [m]	Read table (last page) to TBLH and Data Memory	3 ^{Note}	None
LITABRD [m]	Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory	3 ^{Note}	None
LITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	3 ^{Note}	None
Miscellaneous			
LCLR [m]	Clear Data Memory	2 ^{Note}	None
LSET [m]	Set Data Memory	2 ^{Note}	None
LSWAP [m]	Swap nibbles of Data Memory	2 ^{Note}	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	$ACC \leftarrow ACC + [m] + C$
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	$[m] \leftarrow ACC + [m] + C$
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	$ACC \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC + x$
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	$[m] \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC \text{ "AND" } [m]$
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	$ACC \leftarrow ACC \text{ "AND" } x$
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	$[m] \leftarrow ACC \text{ "AND" } [m]$
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	$[m] \leftarrow [m] - 1$
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	$ACC \leftarrow [m] - 1$
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	$TO \leftarrow 0$ $PDF \leftarrow 1$
Affected flag(s)	TO, PDF
INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	$[m] \leftarrow [m] + 1$
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	$ACC \leftarrow [m] + 1$
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 \leftarrow 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	$ACC \leftarrow [m]$
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	$ACC \leftarrow x$
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	$[m] \leftarrow ACC$
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	$ACC \leftarrow ACC \text{ "OR" } [m]$
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	$ACC \leftarrow ACC \text{ "OR" } x$
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	$[m] \leftarrow ACC \text{ "OR" } [m]$
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 \leftarrow 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 \leftarrow Stack $ACC \leftarrow x$
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 \leftarrow Stack $EMI \leftarrow 1$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim 6)$ $[m].0 \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC \leftarrow ACC - [m] - \bar{C}$
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	$ACC \leftarrow ACC - [m] - \bar{C}$
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	$[m] \leftarrow ACC - [m] - \bar{C}$
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	$[m] \leftarrow [m] - 1$ 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	$ACC \leftarrow [m] - 1$ 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	$[m] \leftarrow FFH$
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	$[m].i \leftarrow 1$
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	$[m] \leftarrow [m] + 1$ 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	$ACC \leftarrow [m] + 1$ 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 $[m].i \neq 0$
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 $[m] \neq 0$
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	$ACC \leftarrow ACC - [m]$
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	$[m] \leftarrow ACC - [m]$
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	$ACC \leftarrow ACC - x$
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	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
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	$ACC \leftarrow [m]$ 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	$ACC \leftarrow ACC + [m] + C$
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	$[m] \leftarrow ACC + [m] + C$
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	$ACC \leftarrow ACC + [m]$
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	$[m] \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC \text{ "AND" } [m]$
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	$[m] \leftarrow ACC \text{ "AND" } [m]$
Affected flag(s)	Z
LCLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	$[m] \leftarrow 00H$
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	$[m].i \leftarrow 0$
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	$ACC \leftarrow [m]$
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	$[m] \leftarrow ACC$
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	$ACC \leftarrow ACC \text{ "OR" } [m]$
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	$[m] \leftarrow ACC \text{ "OR" } [m]$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
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	$[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow [m].0$
Affected flag(s)	None
LRRRA [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	$ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow [m].0$
Affected flag(s)	None
LRRRC [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	$[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
LSBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
LSBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m] - \bar{C}$
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	$[m] \leftarrow [m] - 1$ 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	$ACC \leftarrow [m] - 1$ 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	$[m] \leftarrow FFH$
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	$[m].i \leftarrow 1$
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	$[m] \leftarrow [m] + 1$ 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	$ACC \leftarrow [m] + 1$ 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 $[m].i \neq 0$
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

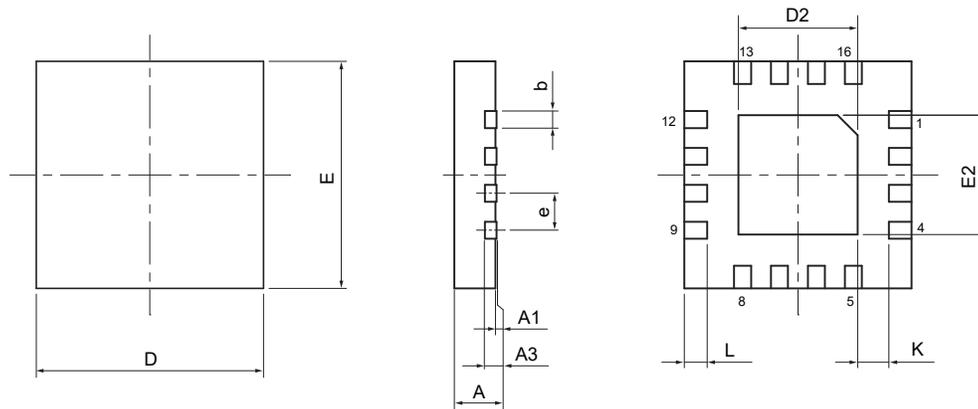
Package Information

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

SAW Type 16-pin QFN (3mm×3mm×0.75mm, FP0.25mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	0.008 REF		
b	0.007	0.010	0.012
D	0.118 BSC		
E	0.118 BSC		
e	0.020 BSC		
D2	0.063	—	0.069
E2	0.063	—	0.069
L	0.008	0.010	0.012
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	0.203 REF		
b	0.18	0.25	0.30
D	3.00 BSC		
E	3.00 BSC		
e	0.50 BSC		
D2	1.60	—	1.75
E2	1.60	—	1.75
L	0.20	0.25	0.30
K	0.20	—	—

Copyright© 2025 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

The information provided in this document has been produced with reasonable care and attention before publication, however, HOLTEK does not guarantee that the information is completely accurate. The information contained in this publication is provided for reference only and may be superseded by updates. HOLTEK disclaims any expressed, implied or statutory warranties, including but not limited to suitability for commercialization, satisfactory quality, specifications, characteristics, functions, fitness for a particular purpose, and non-infringement of any third-party's rights. HOLTEK disclaims all liability arising from the information and its application. In addition, HOLTEK does not recommend the use of HOLTEK's products where there is a risk of personal hazard due to malfunction or other reasons. HOLTEK hereby declares that it does not authorize the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.