Continuous Glucose Monitoring Flash MCU

BH66F2475

Table of Contents

Features

CPU Features

- Operating Voltage
  - f_{SYS}=4MHz: 2.2V~5.5V
  - f_{SYS}=8MHz: 2.2V~5.5V
  - f_{SYS}=12MHz: 2.7V~5.5V
- Up to 0.33μs instruction cycle with 12MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator Types
  - Internal High Speed 4/8/12MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Fully integrated internal oscillators require no external components
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 16-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 32K×16
- Data Memory: 2048×8
- True EEPROM Memory: 2048×8
- In Application Programming function – IAP
- Internal On-Chip Debug Support function – OCDS
- Watchdog Timer function
- 9 bidirectional I/O lines
- 4 external interrupt lines shared with I/O pins
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
- Dual Time Base functions for generation of fixed time interrupt signals
- Continuous Glucose Monitoring AFE circuit
  - Multi-channel 24-bit resolution Delta Sigma A/D Converter
  - Internal Reference Voltage Generator
  - Four 12-bit D/A Converters
  - 3 internal Operational Amplifiers
  - 1 comparator function
  - Internal LDO
- Serial Interface Module – SIM for SPI or I²C interface
- Integrated Multiplier/Divider Unit – MDU
- Integrated 16-bit Cyclic Redundancy Check function – CRC
- Low voltage reset function
- Package types: 16/24-pin QFN

General Description

The device is a Flash Memory 8-bit high performance RISC architecture microcontroller device with Continuous Glucose Monitoring (CGM) AFE module that specifically designed for CGM applications.

For memory features, the Flash Memory offers users the convenience of Flash Memory 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, four 12-bit D/A converters, three internal operational amplifiers and one comparator. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI and I²C interface functions, three popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

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

With regard to CGM applications, the device has integrated many of the functions required by these products. These include functions such as Internal Reference Voltage generator, Internal LDO, 24-bit Delta Sigma A/D converter, 12-bit D/A Converters and operational amplifiers, etc. The inclusion of flexible I/O programming features, 16-bit MDU, 16-bit Cyclic Redundancy Check function, Time Base functions along with many other features enhance the versatility of the device to suit for CGM applications.

Block Diagram

Pin Assignment

16 QFN-A pin assignment for BH66F2475:

Top pins (13-16): OP1O, DAC2O, OP2N, OP2O

- OP1N — 1
- DAC1O — 2
- DACVREF — 3
- AVSS — 4
- 12 — PA0/OCDSDA/ICPDA/SDO
- 11 — PA2/OCDSCK/ICPCK/SDI/SDA
- 10 — PA1/AIN1
- 9 — PA3/AIN0

Bottom pins (5-8): VOREG, VDD/AVDD, VSS, PB0/INT3/SCK/SCL/PTP/CMF

24 QFN-A pin assignment for BH66F2475:

Top pins (19-24): OP1N, OP1O, DAC2O, VSS, OP2N, OP2O

- DAC1O — 1
- OP3O — 2
- OP3N — 3
- DAC3O — 4
- DACVREF — 5
- AVSS — 6
- 18 — PA4/INT0/CTCK0/CTP0/SCK/SCL
- 17 — PA0/OCDSDA/ICPDA/SDO
- 16 — PA2/OCDSCK/ICPCK/SDI/SDA
- 15 — PA1/AIN1
- 14 — PA3/AIN0
- 13 — PA5/INT1/PTCK/SCS/CMPO

Bottom pins (7-12): VOREG, VDD/AVDD, VSS, PB0/NT3/SCK/SCL/PTP/CMPO, PA7/INT2/SDI/SDA/PTPI, PA6/CTCK1/CTP1/SDO

Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.

2. The OCDSDA and OCDSCK pins are used as the OCDS dedicated pins.

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

Pin Descriptions

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

Pin Name	Function	OPT	I/T	O/T	Description
PA0/OCDSDA/ICPDA/ SDO	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OCDSDA	—	ST	CMOS	OCDS data/address pin
	ICPDA	—	ST	CMOS	ICP data/address pin
	SDO	PAS0	—	CMOS	SPI serial data output
PA1/AIN1	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	AIN1	PAS0	AN	—	A/D Converter analog input
PA2/OCDSCK/ICPCK/ SDI/SDA	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OCDSCK	—	ST	—	OCDS clock pin
	ICPCK	—	ST	—	ICP clock pin
	SDI	PAS0 IFS	ST	—	SPI serial data input
	SDA	PAS0 IFS	ST	NMOS	I²C data line
PA3/AIN0	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	AIN0	PAS0	AN	—	A/D Converter analog input
PA4/INT0/CTCK0/ CTP0/SCK/SCL	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT0	PAS1 INTEG INTC0	ST	—	External interrupt
	CTCK0	PAS1	ST	—	CTM0 clock input
	CTP0	PAS1	—	CMOS	CTM0 output
	SCK	PAS1 IFS	ST	CMOS	SPI serial clock
	SCL	PAS1 IFS	ST	NMOS	I²C clock line
PA5/INT1/PTCK/$\overline{\text{SCS}}$/ CMPO	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT1	PAS1 INTEG INTC0	ST	—	External interrupt
	PTCK	PAS1	ST	—	PTM clock input or capture input
	$\overline{\text{SCS}}$	PAS1	ST	CMOS	SPI slave select pin
	CMPO	PAS1	—	CMOS	Comparator output

Pin Name	Function	OPT	I/T	O/T	Description
PA6/CTCK1/CTP1/ SDO	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK1	PAS1	ST	—	CTM1 clock input
	CTP1	PAS1	—	CMOS	CTM1 output
	SDO	PAS1	—	CMOS	SPI serial data output
PA7/INT2/SDI/SDA/PTPI	PA7	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT2	PAS1 INTEG INTC1	ST	—	External interrupt
	SDI	PAS1 IFS	ST	—	SPI serial data input
	SDA	PAS1 IFS	ST	NMOS	I²C data line
	PTPI	PAS1	ST	—	PTM capture input
PB0/INT3/SCK/SCL/ PTP/CMPO	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	INT3	PBS0 INTEG INTC2	ST	—	External interrupt
	SCK	PBS0 IFS	ST	CMOS	SPI serial clock
	SCL	PBS0 IFS	ST	NMOS	I²C clock line
	PTP	PBS0	—	CMOS	PTM output
	CMPO	PBS0	—	CMOS	Comparator output
VOREG	VOREG	—	—	AN	LDO regulator output
		—	AN	—	Positive power supply for PGA, A/D converter
OP1N	OP1N	—	AN	—	OPA1 negative input
OP1O	OP1O	—	—	AN	OPA1 output
OP2N	OP2N	—	AN	—	OPA2 negative input
OP2O	OP2O	—	—	AN	OPA2output
OP3N	OP3N	—	AN	—	OPA3 negative input
OP3O	OP3O	—	—	AN	OPA3 output
DACVREF	DACVREF	—	—	AN	D/A Converter reference voltage
DAC1O	DAC1O	—	—	AN	D/A Converter output
DAC2O	DAC2O	—	—	AN	D/A Converter output
DAC3O	DAC3O	—	—	AN	D/A Converter output
VDD/AVDD	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
VSS	VSS	—	PWR	—	Digital negative power supply
AVSS	AVSS	—	PWR	—	Analog negative power supply

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

Absolute Maximum Ratings

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

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

Storage Temperature.. -60°C to 150°C

Operating Temperature.. -40°C~85°C

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

I_{OL} Total 80mA

Total Power Dissipation 500mW

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

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=-40°C~85°C

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

Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions V_{DD}	Conditions	Min.	Typ.	Max.	Unit
I_{DD}	SLOW Mode – LIRC	2.2V	f_{SYS}=32kHz	—	8	16	µA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	2.2V	f_{SYS}=4MHz	—	0.3	0.5	mA
		3V		—	0.4	0.6	
		5V		—	0.8	1.2	
		2.2V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	
		2.7V	f_{SYS}=12MHz	—	1.0	1.4	mA
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	

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

1. Any digital inputs are setup in a non floating condition.

2. All measurements are taken under conditions of no load and with all peripherals in an off state.

3. There are no DC current paths.

4. All Operating Current values are measured using a continuous NOP instruction program loop.

Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.2V	WDT off	—	0.45	0.80	7.00	μA
		3V		—	0.45	0.90	8.00	
		5V		—	0.5	2.0	10.0	
		2.2V	WDT on	—	1.5	3.0	3.6	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3	5	6	
	IDLE0 Mode – LIRC	2.2V	f_{SUB} on	—	3	5	6	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
I_{STB}	IDLE1 Mode – HIRC	2.2V	f_{SUB} on, f_{SYS}=4MHz	—	144	200	240	μA
		3V		—	180	250	300	
		5V		—	400	600	720	
		2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	540	750	900	
		5V		—	800	1200	1440	

Note: When using the characteristic table data, the following notes should be taken into consideration:
1. Any digital inputs are setup in a non floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

A.C. Characteristics

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

Internal High Speed Oscillator – HIRC – Frequency Accuracy

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

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

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

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

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

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

Internal Low Speed Oscillator Characteristics – LIRC

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

Operating Frequency Characteristic Curves

System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time (Wake-up from Conditions where f_{SYS} is off)	—	$f_{SYS}=f_H~f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
			$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time (Wake-up from Conditions 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}

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{RSTD}	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR_{POR}=5V/ms	14	16	18	ms
	System Reset Delay Time (LVRC/WDTC/RSTC Software Reset)	—	—				
	System Reset Delay Time (Reset Source from WDT Overflow)	—	—	14	16	18	
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	µs

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

2. The time units, shown by the symbols t_{HIRC} etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example, $t_{HIRC}=1/f_{HIRC}$, $t_{SYS}=1/f_{SYS}$ etc.

3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.

4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

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[1]	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I_{LEAK}	Input leakage current for I/O Ports	3V	V_{IN}=V_{DD} or V_{IN}=V_{SS}	—	—	±1	µA
		5V		—	—	±1	
t_{INT}	External Interrupt Input Minimum Pulse Width	—	—	10	—	—	µs
t_{TPI}	PTPI Input Pin Minimum Pulse Width	—	—	0.3	—	—	µs
t_{TCK}	xTCK Input Pin Minimum Pulse Width	—	—	0.3	—	—	µs
f_{TMCLK}	PTM Maximum Timer Clock Source Frequency	5V	—	—	—	1	f_{SYS}
t_{CPW}	PTM Minimum Capture Pulse Width	—	—	t_{CPW}[2]	—	—	µs

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

2. $t_{TMCLK}=1/f_{TMCLK}$

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

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

Ex1: If PTCAPTS=0, $f_{TMCLK}=8MHz$, $t_{TPI}=0.3µs$, then $t_{CPW}=max(0.125µs, 0.3µs)=0.3µs$

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

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

Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Flash Program Memory							
t_{FER}	Erase Time	—	FWERTS=0	—	3.2	3.9	ms
		—	FWERTS=1	—	3.7	4.5	
t_{FWR}	Write Time	—	FWERTS=0	—	2.2	2.7	ms
		—	FWERTS=1	—	3.0	3.6	
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
t_{ACTV}	ROM Activation Time – Wake-up from IDLE/SLEEP Mode	—	—	32	—	64	μs
Data EEPROM Memory							
V_{DD}	V_{DD} for Read / Write	—	—	2.2	—	5.5	V
t_{EERD}	EEPROM Read Time	—	—	—	—	4	t_{SYS}
t_{EEER}	EEPROM Erase Time	—	EWERTS=0	—	3.2	3.9	ms
		—	EWERTS=1	—	3.7	4.5	
t_{EEWR}	EEPROM Write Time (Byte Mode)	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	
	EEPROM Write Time (Page Mode)	—	EWERTS=0	—	2.2	2.7	
		—	EWERTS=1	—	3.0	3.6	
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

Note: 1. "E/W" means Erase/Write times.

2. The ROM activation time t_{ACTV} should be added when calculating the total system start-up time of a wake-up from the IDLE/SLEEP mode.

LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	-3%	2.10	+3%	V
		—	LVR enable, voltage select 2.55V		2.55		
		—	LVR enable, voltage select 3.15V		3.15		
		—	LVR enable, voltage select 3.8V		3.80		
I_{LVROP}	LVR Operating Current	3V	LVR enable, V_{LVR}=2.1V	—	—	10	μA
		5V		—	10	15	
t_{LVR}	Minimum Low Voltage Width to Reset	—	TLVR[1:0]=00B	120	240	480	μs
			TLVR[1:0]=01B	0.5	1.0	2.0	ms
			TLVR[1:0]=10B	1	2	4	
			TLVR[1:0]=11B	2	4	8	
I_{LVR}	Additional Current for LVR Enable	5V	—	—	—	14	μA

Analog Front End Circuit Characteristics

LDO Electrical Characteristics

Ta=0°C~50°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				
AV_{DD}	LDO Input Voltage	—	—	2.2	—	5.5	V
I_Q	LDO Quiescent Current	—	LDOVS[1:0]=00B, AV_{DD}=3.6V, no load	—	25	—	µA
V_{OUT_LDO}	LDO Output Voltage	—	LDOVS[1:0]=00B, AV_{DD}=3.6V, I_{LOAD}=0.1mA	-5%	2.4	+5%	V
		—	LDOVS[1:0]=01B, AV_{DD}=3.6V, I_{LOAD}=0.1mA		2.6		
		—	LDOVS[1:0]=10B, AV_{DD}=3.6V, I_{LOAD}=0.1mA		2.2		
		—	LDOVS[1:0]=11B, AV_{DD}=3.6V, I_{LOAD}=0.1mA		2.3		
ΔV_{LOAD}	LDO Load Regulation	—	LDOVS[1:0]=00B, AV_{DD}=V_{OUT_LDO}+0.2V, 0mA≤I_{LOAD}≤10mA	—	0.105	0.210	%/mA
V_{DROP_LDO}	LDO Dropout Voltage	—	LDOVS[1:0]=00B, AV_{DD}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}= 2%	—	—	220	mV
		—	LDOVS[1:0]=01B, AV_{DD}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}=2%	—	—	200	
		—	LDOVS[1:0]=10B, AV_{DD}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}=2%	—	—	240	
		—	LDOVS[1:0]=11B, AV_{DD}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}= 2%	—	—	230	
TC_{LDO}	LDO Temperature Coefficient	—	Ta=0°C~50°C, LDOVS[1:0]=00B, AV_{DD}=3.6V, I_{LOAD}=100µA	—	—	±200	ppm/°C
ΔV_{LINE_LDO}	LDO Line Regulation	—	LDOVS[1:0]=00B, 2.6V≤ AV_{DD} ≤5.5V, I_{LOAD}=100µA	—	—	1	%/V
		—	LDOVS[1:0]=00B, 2.6V≤ AV_{DD} ≤3.6V, I_{LOAD}=100µA	—	—	0.7	%/V

Operational Amplifier Electrical Characteristics

Ta=0°C~50°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{OREG}	Supply Voltage for OPA	—	—	2.2	—	5.5	V
I_{OPA}	Additional Current for OPA	—	No load	—	0.4	—	µA
V_{OS}	Input Offset Voltage	—	—	-15	—	15	mV
I_{OS}	Input Offset Current	—	—	—	1	—	pA
I_B	Input Bias Current	—	—	—	1	—	pA
V_{CM_OPA}	OPA Common Mode Voltage Range	—	—	V_{SS}	—	V_{OREG}	V
PSRR	Power Supply Rejection Ratio	—	—	—	80	—	dB
CMRR	Common Mode Rejection Ratio	—	—	—	80	—	dB

Internal Reference Voltage Characteristics

Ta=0°C~50°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IREF}	Internal Reference Voltage	3V	IREFEN=1, PVREF=10000000B	-3%	1.25	+3%	V
I_{IREF}	Additional Current for Internal Reference Voltage	—	IREFEN=1	—	1.6	—	μA
TC_{IREF}	Temperature Coefficient of Internal Reference Voltage	3V	Ta=0°C~50°C	—	±40	—	ppm/°C

12-bit D/A Converter Electrical Characteristics

Ta=0°C~50°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{OREG}	Supply Voltage for DAC	—	—	2.2	—	5.5	V
I_{DAC}	Additional Current for DAC	—	—	—	1.2	—	μA
DNL	Differential Non-linearity	3V	DACVRS[1:0]=00B	—	—	±8	LSB
INL	Integral Non-linearity	3V	DACVRS[1:0]=00B	—	—	±20	LSB
R_o	R2R Output Resistance	3V	—	—	5	—	kΩ
V_{DACO}	Output Voltage Range	—	—	0.00	—	1.00	$V_{DACVREF}$

A/D Converter Electrical Characteristics

Ta=0°C~50°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{OREG}	Supply Voltage for A/D, PGA	—	LDOEN=0	2.2	—	2.6	V
		—	LDOEN=1	2.2	—	2.6	V
I_{ADC}	Additional Current for ADC Enable	—	—	—	400	—	μA
I_{ADSTB}	Standby Current	—	MCU enter SLEEP mode, no load	—	—	1	μA
N_R	Resolution	—	—	—	—	24	Bit
INL	Integral Non-linearity	—	V_{OREG}=2.6V, V_{REF}=1.25V, ΔSI=±450mV, PGAGN=1	—	±50	—	ppm
NFB	Noise Free Bits	—	f_{MCLK}=4MHz, FLMS[2:0]=000B, V_{OREG}=2.6V, V_{REF}=1.25V, PGAGN=128, OSR=16384	—	16	—	Bit
ENOB	Effective Number of Bits	—	f_{MCLK}=4MHz, FLMS[2:0]=000B, V_{OREG}=2.6V, V_{REF}=1.25V, PGAGN=128, OSR=16384	—	18.7	—	Bit
f_{ADCK}	ADC Clock Frequency	—	—	40.0	409.6	440.0	kHz
f_{ADO}	ADC Output Data Rate	—	f_{MCLK}=4MHz, FLMS[2:0]=000B SINC3[1:0]=11B	4	—	1042	Hz
		—	f_{MCLK}=4MHz, FLMS[2:0]=010B SINC3[1:0]=11B	10	—	2604	Hz
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	V

Effective Number of Bits – ENOB

$AV_{DD}=V_{OREG}=2.4V$, $V_{REF}=1.25V$, $f_{ADCK}=133kHz$

Data Rate (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
4	21.3	21.3	21.2	21.1	21.0	20.4	19.8	18.4
8	20.9	20.9	20.9	20.8	20.5	20.1	19.0	18.3
16	19.2	19.2	19.1	19.0	18.9	18.7	18.9	18.0
33	18.9	18.9	18.9	18.9	18.7	18.6	18.3	17.4
65	18.8	18.8	18.8	18.7	18.6	18.3	17.7	16.9
130	18.5	18.5	18.5	18.4	18.4	18.0	17.4	16.5
260	18.4	18.3	18.3	18.3	18.1	17.5	16.8	15.9
521	17.6	17.6	17.5	17.5	17.5	17.2	16.3	15.4

$AV_{DD}=V_{OREG}=2.4V$, $V_{REF}=1.25V$, $f_{ADCK}=333kHz$

Data Rate (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
10	21.0	20.9	20.9	20.8	20.6	19.9	19.2	18.2
20	20.7	20.6	20.6	20.5	20.2	19.6	18.7	17.6
41	20.5	20.5	20.5	20.3	19.7	18.5	18.0	17.4
81	19.4	19.3	19.3	19.1	18.8	18.2	17.6	16.1
163	18.6	18.6	18.2	17.8	17.0	16.1	15.1	14.1
326	18.2	18.0	17.7	17.3	16.4	15.5	14.5	13.5
651	18.1	18.1	18.0	17.9	17.5	17.1	16.2	15.3
1302	17.8	17.8	17.6	17.6	17.2	16.6	15.8	15.0

PGA Electrical Characteristics

Ta=0°C~50°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
ΔD_I	Differential Input Voltage Range	—	Gain=PGAGN×ADGN	$-V_{REF}/Gain$	—	$+V_{REF}/Gain$	V

I²C Electrical Characteristics

Ta=25°C

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_DD	Conditions				
$t_{SCL(L)}$	SCL Clock Low Time	3V/5V	Standard mode	3.5	—	—	µs
			Fast mode	0.9	—	—	
t_{FALL}	SCL and SDA Fall Time	3V/5V	Standard mode	—	—	1.3	µs
			Fast mode	—	—	0.34	
t_{RISE}	SCL and SDA Rise Time	3V/5V	Standard mode	—	—	1.3	µs
			Fast mode	—	—	0.34	
$t_{SU(SDA)}$	SDA Data Setup Time	3V/5V	Standard mode	0.25	—	—	µs
			Fast mode	0.1	—	—	
$t_{H(SDA)}$	SDA Data Hold Time	3V/5V	—	0.1	—	—	µs
$t_{VD(SDA)}$	SDA Data Valid Time	3V/5V	—	—	—	0.6	µs
$t_{SU(STA)}$	Start Condition Setup Time	3V/5V	Standard mode	3.5	—	—	µs
			Fast mode	0.6	—	—	
$t_{H(STA)}$	Start Condition Hold Time	3V/5V	Standard mode	4.0	—	—	µs
			Fast mode	0.6	—	—	
$t_{SU(STO)}$	Stop Condition Setup Time	3V/5V	Standard mode	3.5	—	—	µs
			Fast mode	0.6	—	—	

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

I²C Timing Diagram
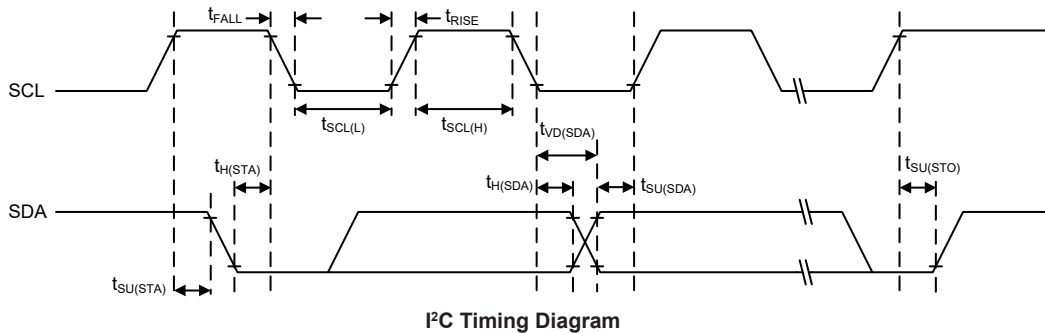
Power-on Reset Characteristics

Ta=-40°C~85°C

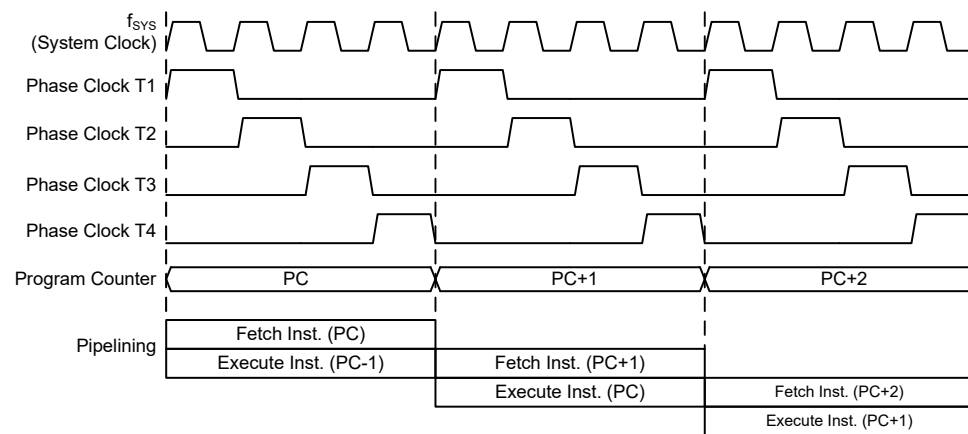| Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_DD	Conditions				
V_{POR}	V_DD Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR_{POR}	V_DD Rising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for V_DD Stays at V_POR to Ensure Power-on Reset	—	—	1	—	—	ms
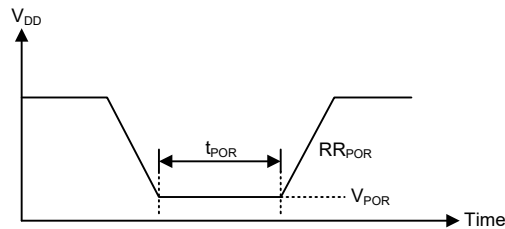
System Architecture

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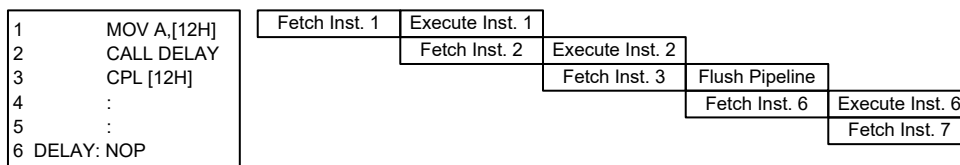
Clocking and Pipelining

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.

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

System Clocking and Pipelining

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

Fetch Inst. 1	Execute Inst. 1
Fetch Inst. 2	Execute Inst. 2
Fetch Inst. 3	Flush Pipeline
Fetch Inst. 6	Execute Inst. 6
Fetch Inst. 7

Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demand a jump to a non-consecutive Program Memory address. For the device with a Program Memory capacity in excess of 8K words, the Program Memory address may be located in a certain program memory bank which is selected by the program memory bank pointer bits, PBP1~PBP0. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

| Program Counter	
High Byte	**Low Byte (PCL)**
PBP1~PBP0, PC12~PC8	PCL7~PCL0

Program Counter

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

Program Counter Read Register

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.

Program Counter (Fetch Instruction)
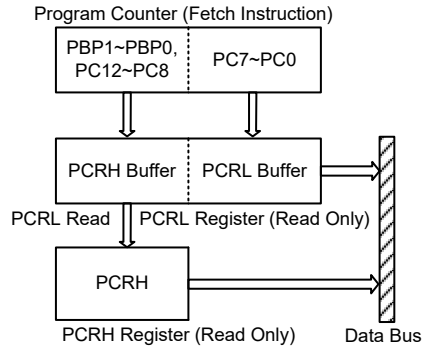
- **PCRL Register**

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**: Low byte register bit 7 ~ bit 0

- **PCRH Register**

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

Bit 7   Unimplemented, read as "0"

Bit 6~0   **D14~D8**: High byte register bit 6 ~ bit 0

Stack

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

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

If the stack is overflow, the first Program Counter save in the stack will be lost.
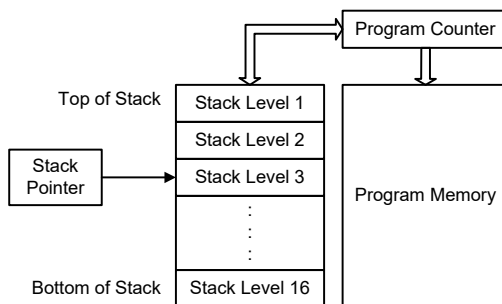
• **STKPTR Register**

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

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

Bit 6~4   Unimplemented, read as "0"

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

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

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

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

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

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

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

Arithmetic and Logic Unit – ALU

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

- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA

- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
  LAND, LOR, LXOR, LANDM, LORM, LXORM, LCPL, LCPLA

- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
  LRRA, LRR, LRRCA, LRRC, LRLA, LRL, LRLCA, LRLC

- Increment and Decrement:
  INCA, INC, DECA, DEC,
  LINCA, LINC, LDECA, LDEC

- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI,
  LSZ, LSZA, LSNZ, LSIZ, LSDZ, LSIZA, LSDZA

Flash Program Memory

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

Structure

The Program Memory has a capacity of 32K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer registers.
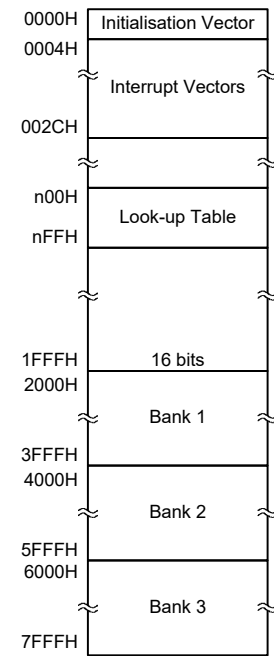
Program Memory Structure

Special Vectors

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

Look-up Table

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

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

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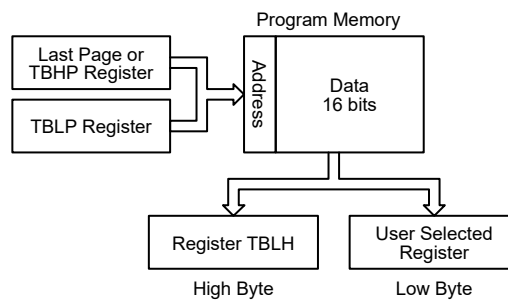

Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is "1F00H" which is located in ROM Bank 3 and refers to the start address of the last page within the 32K words Program Memory. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "7F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by TBHP and TBLP if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

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

Table Read Program Example

```
rombank3 code1
ds .section 'data'
tempreg1 db ?       ; temporary register #1
tempreg2 db ?       ; temporary register #2
code0 .section 'code'
mov a,06H           ; initialise table pointer - note that this address is referenced
mov tblp,a          ; to the last page or the page that tbhp pointed
mov a,7FH           ; initialise high table pointer
mov tbhp,a          ; it is not necessary to set tbhp if executing tabrdl or ltabrdl
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer data at
                    ; program memory address "7F06H" transferred to tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrd tempreg2      ; transfers value in table referenced by table pointer data at
                    ; program memory address "7F05H" transferred to tempreg2 and TBLH
                    ; in this example the data "1AH" is transferred to tempreg1 and
                    ; data "0FH" to tempreg2 the value "00H" will be
                    ; transferred to the high byte register TBLH
:
:
code3 .section 'code'
org 1F00H           ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

In Circuit Programming – ICP

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

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

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

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

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

During the programming process, the user must take care of the ICPDA and ICPCK pins for data and clock programming purposes to ensure that no other outputs are connected to these two pins.
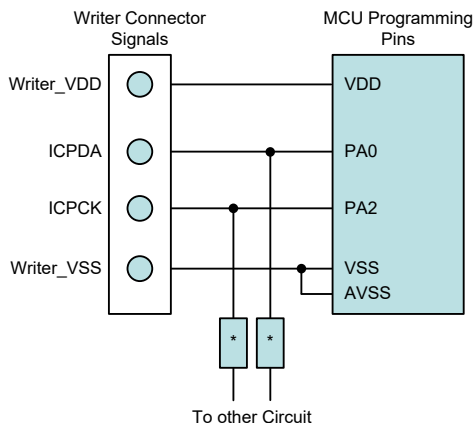
Note: * may be resistor or capacitor. The resistance of * must be greater than 1kΩ or the capacitance of * must be less than 1nF.

On-Chip Debug Support – OCDS

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

Holtek e-Link Pins	MCU Chip Pins	Pin Description
OCDSDA	OCDSDA	On-Chip Debug Support Data/Address input/output
OCDSCK	OCDSCK	On-Chip Debug Support Clock input
VDD	VDD	Power Supply
VSS	VSS&AVSS	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, using I/O pins. Regarding the internal firmware, the user can select versions provided by Holtek or create their own. The following section illustrates the procedures regarding how to implement the IAP firmware.

Flash Memory Read/Write Size

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

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

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

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

IAP Operation Format

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

Page Number and Address Selection
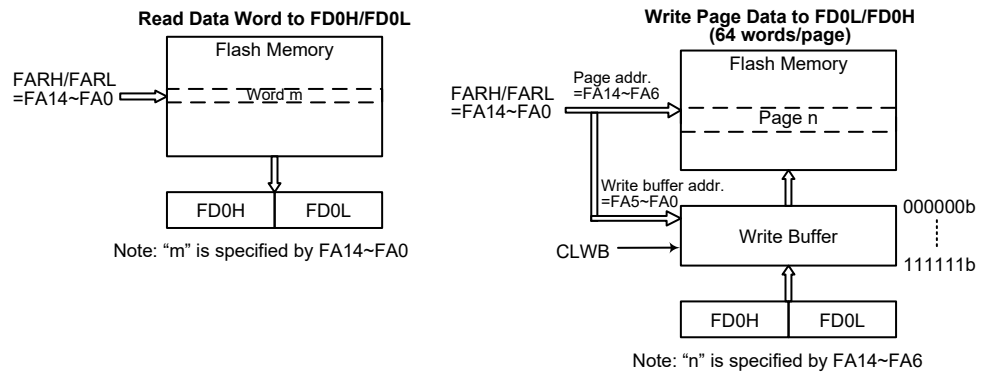
Flash Memory IAP Read/Write Structure

Write Buffer

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

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

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

IAP Flash Program Memory Registers

There are two address registers, four pairs of 16-bit data registers and three control register, which are all located in Sector 1. Read and Write operations to the Flash memory are carried out using 16-bit data operations using the address and data registers and the control registers. Several registers control the overall operation of the internal Flash Program Memory. The address registers are named FARL and FARH, the data registers are named FDnL and FDnH, where n is equal to 0~3, and the control registers are named FC0, FC1 and FC2. As all the IAP related registers are located in Sector 1, they can be addressed directly only using the corresponding extended instructions or can be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pairs and Indirect Addressing Register, IAR1 or IAR2.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	FWERTS	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	FA14	FA13	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Register List

• **FC0 Register**

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

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

        When this bit is cleared to 0 by application program, the Flash memory erase/write function is disabled. Note that writing a "1" into this bit results in no action. This bit is used to indicate the Flash memory erase/write function status. When this bit is set to 1 by hardware, it means that the Flash memory erase/write function is enabled successfully. Otherwise, the Flash memory erase/write function is disabled as the bit is zero.

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

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

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

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

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

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

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

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

Bit 0   **FRD**: Flash memory read initiate control
        0: Do not initiate Flash memory read or indicating that a Flash memory read process has completed
        1: Initiate Flash memory read process

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

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

• **FC1 Register**

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

Bit 7~0   **D7~D0**: Chip Reset Pattern

When a specific value of "55H" is written into this register, a reset signal will be generated to reset the whole chip.

• **FC2 Register**

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

Bit 7~2   Unimplemented, read as "0"

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

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

• **FARL Register**

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

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

• **FARH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	FA14	FA13	FA12	FA11	FA10	FA9	FA8
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7   Unimplemented, read as "0"

Bit 6~0   **FA14~FA8**: Flash Memory Address bit 14 ~ bit 8

• **FD0L Register**

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

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

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

• **FD0H Register**

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

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

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

• **FD1L Register**

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

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

• **FD1H Register**

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

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

• **FD2L Register**

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

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

• **FD2H Register**

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

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

• **FD3L Register**

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

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

• **FD3H Register**

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

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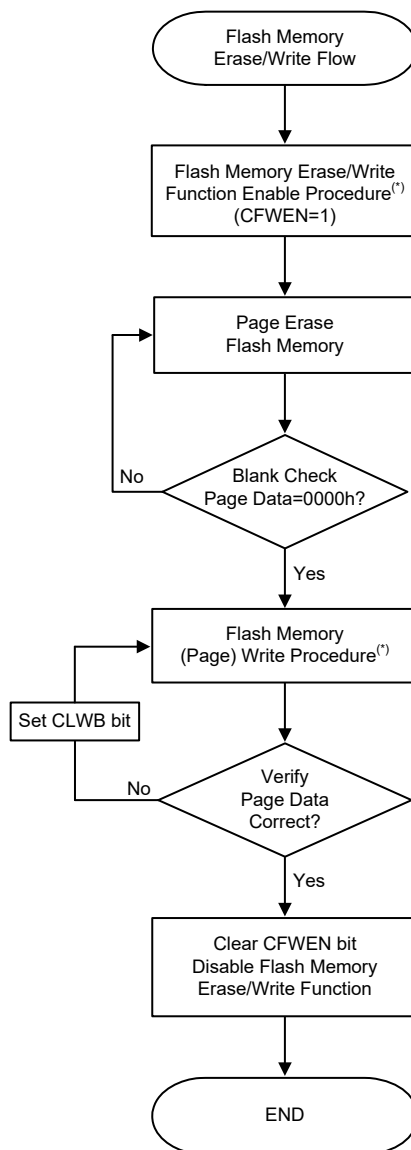
Flash Memory Erase/Write Flow

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

Flash Memory Erase/Write Flow Descriptions

1. Activate the "Flash Memory Erase/Write function enable procedure" first. When the Flash Memory Erase/Write function is successfully enabled, the CFWEN bit in the FC0 register will automatically be set high by hardware. After this, Erase or Write operations can be executed on the Flash memory. Refer to the "Flash Memory Erase/Write Function Enable Procedure" for details.

2. Configure the Flash memory address to select the desired erase page, tag address and then erase this page.

3. For a page erase operation, set the FARL and FARH registers to specify the start address of the erase page, then write dummy data into the FD0H register to tag address. The current address will be internally incremented by one after each dummy data is written into the FD0H register. When the address reaches the page boundary, 111111b, the address will not be further incremented but stop at the last address of the page. Note that the write operation to the FD0H register is used to tag address, it must be implemented to determine which addresses to be erased.

   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 Enable Function. This will also activate an internal timer.

3. Write the correct data pattern into the Flash data registers of FD1L, FD2L, FD3L, FD1H, FD2H and FD3H, successively and as soon as possible after the FWPEN bit is set high. The enable Flash memory erase/write function data pattern is 00H, 0DH, C3H, 04H, 09H and 40H corresponding to the FD1L~FD3L and FD1H~FD3H registers respectively.

4. Once the timer has timed out, the FWPEN bit will automatically be cleared to 0 by hardware regardless of the input data pattern.

5. If the written data pattern is incorrect, the Flash memory erase/write function will not be enabled successfully and the above steps should be repeated. If the written data pattern is correct, the Flash memory erase/write function will be enabled successfully.

6. Once the Flash memory erase/write function is enabled, the Flash memory contents can be updated by executing the page erase and write operations using the IAP control registers.

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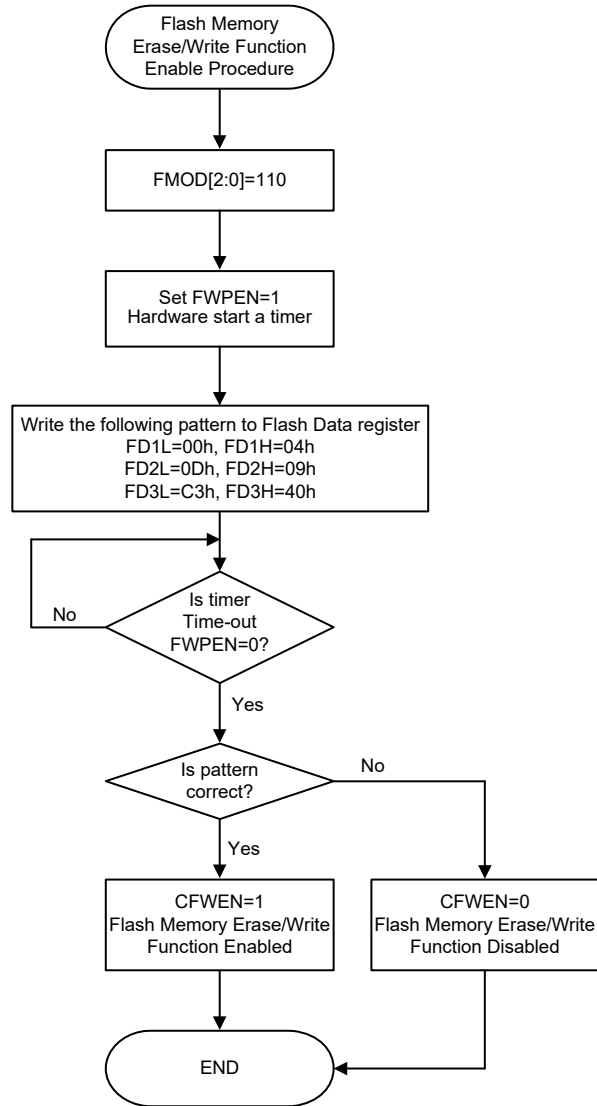
Flash Memory Erase/Write Function Enable Procedure
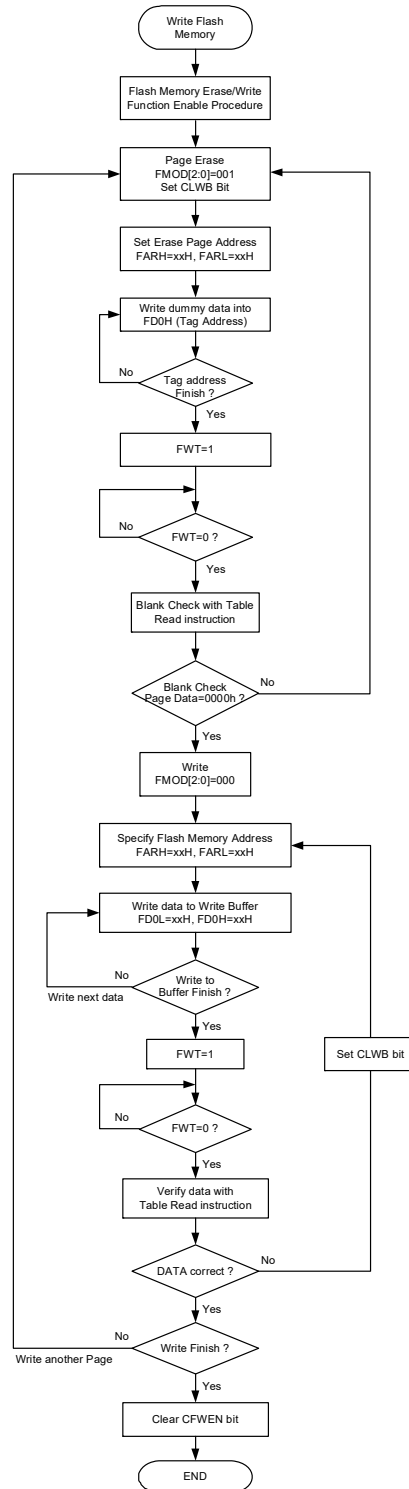
Flash Memory Write Procedure

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

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

Flash Memory Consecutive Write Description

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

1. Activate the "Flash Memory Erase/Write function enable procedure". Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the "Flash Memory Erase/Write function enable procedure" for more details.

2. Set the FMOD2~FMOD0 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 FMOD2~FMOD0 to "000" to select the write operation.

5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L and FD0H registers within a page as specified by their consecutive addresses. The maximum written data number is 64 words.

6. Set the FWT bit high to write the data words from the write buffer to the Flash memory. Wait until the FWT bit goes low.

7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

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

   Go to step 8 if the write operation is successful.

8. Clear the CFWEN bit low to disable the Flash memory erase/write function.

Flash Memory Consecutive Write Procedure

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

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

Flash Memory Non-consecutive Write Description

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

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

1. Activate the "Flash Memory Erase/Write function enable procedure". Check the CFWEN bit value and then execute the erase/write operation if the CFWEN bit is set high. Refer to the "Flash Memory Erase/Write function enable procedure" for more details.

2. Set the FMOD2~FMOD0 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 FMOD2~FMOD0 to "000" to select the write operation.

5. Setup the desired address ADDR1 in the FARH and FRARL 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.
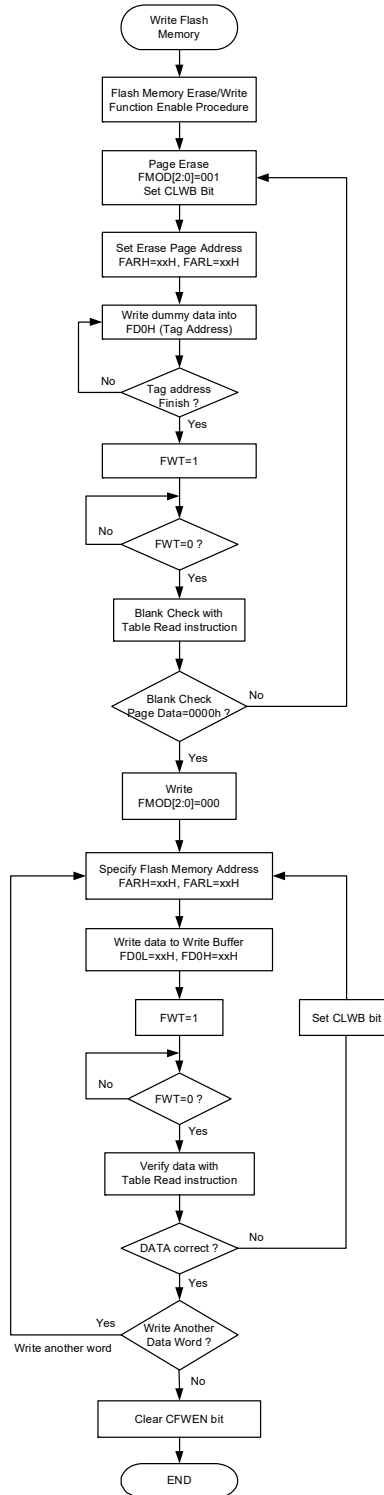
    Go to step 11 if the write operation is successful.

11. Clear the CFWEN bit low to disable the Flash memory erase/write function.
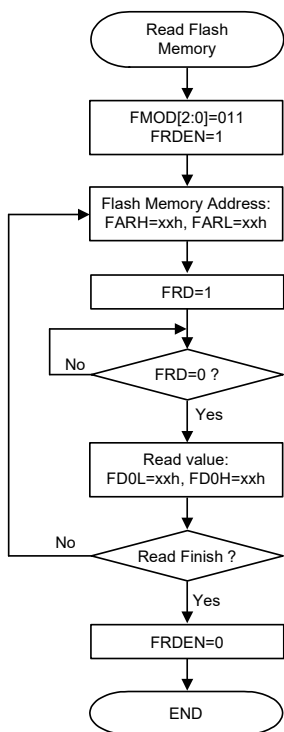
Flash Memory Non-consecutive Write Procedure

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 FMOD2~FMOD0 should be set to "011" to select the Flash memory read mode and the FRDEN bit should be set high to enable the read function. The desired Flash memory address should be written into the FARH and FARL registers and then the FRD bit should be set high. After this the Flash memory read operation will be activated. The data stored in the specified address can be read from the data registers, FD0H and FD0L, when the FRD bit goes low. There is no need to first activate the Flash Memory Erase/Write Function Enable Procedure before the Flash memory read operation is executed.

Flash Memory Read Procedure

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

Data Memory

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

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

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

Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide Memory. Each of the Data Memory Sector is categorized into two types, the Special Purpose Data Memory and the General Purpose Data Memory. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH.

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

Data Memory Summary
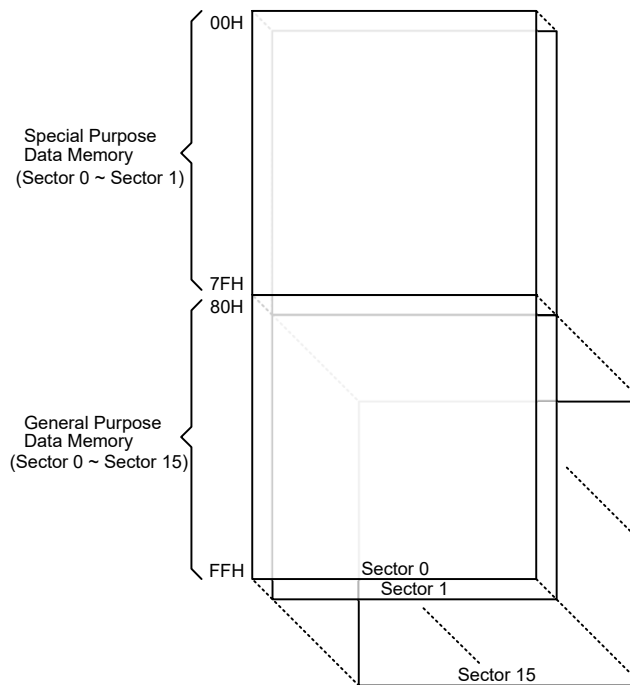
Data Memory Structure

Data Memory Addressing

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

Direct Addressing can be used in all sectors using the corresponding instruction which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address "m" in the extended instructions has 12 valid bits for this device, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH	PBP	
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	SCC	
11H	HIRCC	
12H	STKPTR	
13H	IECC	
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	RSTC	
19H	LVRC	
1AH	TLVRC	
1BH	MFI0	
1CH	MFI1	
1DH		
1EH	WDTC	
1FH	INTEG	
20H	INTC0	
21H	INTC1	
22H	INTC2	
23H	INTC3	
24H	PB	
25H	PBC	
26H	PBPU	
27H	PCRL	
28H	PCRH	
29H		
2AH		
2BH		
2CH	PSCR	
2DH	TB0C	
2EH	TB1C	
2FH	(Reserved)	
30H	(Reserved)	CTM0C0
31H	(Reserved)	CTM0C1
32H	(Reserved)	CTM0DL
33H	(Reserved)	CTM0DH
34H	(Reserved)	CTM0AL
35H	(Reserved)	CTM0AH
36H	(Reserved)	CTM1C0
37H	(Reserved)	CTM1C1
38H	PTMC0	CTM1DL
39H	PTMC1	CTM1DH
3AH	PTMDL	CTM1AL
3BH	PTMDH	CTM1AH
3CH	PTMAL	CRCCR
3DH	PTMAH	CRCIN
3EH	PTMRPL	CRCDL
3FH	PTMRPH	CRCDH

	Sector 0	Sector 1
40H		EEC
41H	EEAL	
42H	EEAH	
43H	EED	FC0
44H	PWRC	FC1
45H	IREFC	FC2
46H	PVREF	FARL
47H	OPA1C	FARH
48H	OPA2C	FD0L
49H	OPA3C	FD0H
4AH	GSC1	FD1L
4BH	AFEDA1C	FD1H
4CH	AFEDA1L	FD2L
4DH	AFEDA1H	FD2H
4EH	AFEDA2C	FD3L
4FH	AFEDA2L	FD3H
50H	AFEDA2H	IFS
51H	AFEDA3C	
52H	AFEDA3L	
53H	AFEDA3H	PAS0
54H	AFEDA0C	PAS1
55H	AFEDA0L	PBS0
56H	AFEDA0H	
57H		
58H		
59H	MDUWR0	
5AH	MDUWR1	
5BH	MDUWR2	
5CH	MDUWR3	
5DH	MDUWR4	
5EH	MDUWR5	
5FH	MDUWCTRL	
60H	PGAC0	
61H	PGAC1	
62H	PGACS	
63H	ADRL	
64H	ADRM	
65H	ADRH	
66H	ADCR0	
67H	ADCR1	
68H	SINC3	
69H	ADCR2	
6AH	ADCR3	
6BH	ADCS	
6CH	(Reserved)	
6DH	CMPC	
6EH	PRFC	
6FH	TDAVRST	
70H	TDAVRLEN	
71H	TOPST	
72H	TOPLEN	
73H	TDAST	
74H	TDALEN	
75H	TCPST	
76H	TCPLEN	
77H	TCHS0	
78H	TCHS1	
79H		
7AH	SIMC0	
7BH	SIMC1	
7CH	SIMD	
7DH	SIMA/SIMC2	
7EH	SIMTOC	
7FH	(Reserved)	

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

Special Purpose Data Memory Structure

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, IAR2

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

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

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

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

Indirect Addressing Program Example

Example 1

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

Example 2

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

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

Direct Addressing Program Example using extended instructions

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

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

Program Memory Bank Pointer – PBP

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

• **PBP Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PBP1	PBP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2   Unimplemented, read as "0"

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

Accumulator – ACC

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

Program Counter Low 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.

Status Register – STATUS

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

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

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

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.

- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.

- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.

- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.

- PDF is cleared by a system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction.

- TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.

- SC is the result of the "XOR" operation which is performed by the OV flag and the MSB of the current instruction operation result.

- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.

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

- **STATUS Register**

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

"x": unknown

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

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

Bit 5　　**TO**: Watchdog Time-out flag

　　　　0: After power up or executing the "CLR WDT" or "HALT" instruction
　　　　1: A watchdog time-out occurred

Bit 4　　**PDF**: Power down flag

　　　　0: After power up or executing the "CLR WDT" instruction
　　　　1: By executing the "HALT" instruction

Bit 3　　**OV**: Overflow flag

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

Bit 2　　**Z**: Zero flag

　　　　0: The result of an arithmetic or logical operation is not zero
　　　　1: The result of an arithmetic or logical operation is zero

Bit 1　　**AC**: Auxiliary flag

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

Bit 0　　**C**: Carry flag

　　　　0: No carry-out
　　　　1: An operation results in a carry during an addition operation or if a borrow does
　　　　　 not take place during a subtraction operation
　　　　The "C" flag is also affected by a rotate through carry instruction.

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 2048×8 bits for this device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using a pair of address registers and a data register in Sector 0 and a single control register in Sector 1.

EEPROM Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEAL	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH	—	—	—	—	—	EEAH2	EEAH1	EEAH0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD

EEPROM Register List

• EEAL Register

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

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

• EEAH Register

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

Bit 7~3   Unimplemented, read as "0"

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

• EED Register

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

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

• EEC Register

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

Bit 7   **EWERTS**: Data EEPROM Erase time and Write time select
        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 the hardware after the erase cycle has finished. Clearing this bit to zero will inhibit data EEPROM erase operations.

Bit 5   **ER**: Data EEPROM erase control

    0: Erase cycle has finished

    1: Activate an erase cycle

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

Bit 4   **MODE**: Data EEPROM operation mode selection

    0: Byte operation mode

    1: Page operation mode

This is the EEPROM operation mode selection bit. When the bit is set high by the application program, the Page write, erase or read function will be selected. Otherwise, the byte write or read function will be selected. The EEPROM page buffer size is 16-byte.

Bit 3   **WREN**: Data EEPROM write enable

    0: Disable

    1: Enable

This is the Data EEPROM Write Enable bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations. Note that the WREN bit will automatically be cleared to zero after the write operation is finished.

Bit 2   **WR**: Data EEPROM write control

    0: Write cycle has finished

    1: Activate a write cycle

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

Bit 1   **RDEN**: Data EEPROM read enable

    0: Disable

    1: Enable

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

Bit 0   **RD**: Data EEPROM read control

    0: Read cycle has finished

    1: Activate a read cycle

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

Note: 1. The EREN, ER, WREN, WR, RDEN and RD cannot be set to "1" at the same time in one instruction.

    2. Ensure that the f_{SUB} clock is stable before executing the erase or write operation.

    3. Ensure that the erase or write operation is totally complete before changing contents of the EEPROM related registers or activating the IAP function.

Read Operation from the EEPROM

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

Byte Read Mode

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

The EEPROM address higher 7 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page read operation mode the lower 4-bit address value will automatically be incremented by one. However, the higher 7-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not "roll over".

Page Erase Operation to the EEPROM

The EEPROM page erase operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page erase. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM erase enable control bit, namely EREN, is changed from "1" to "0", the internal page buffer will also be cleared. Note that when the EREN bit is changed from "0" to "1", the internal page buffer will not be cleared. The EEPROM address higher 7 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page erase operation mode the lower 4-bit address value will automatically be incremented by one after each dummy data byte is written into the EED register. However, the higher 7-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not "roll over".

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

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, indicating that the page data has been erased. The application program can therefore poll the ER bit to determine when the erase cycle has ended. After the erase operation is finished, the EREN bit will be cleared to zero by hardware. The Data EEPROM erased page content will all be zero after a page erase operation.

Write Operation to the EEPROM

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

Byte Write Mode

The EEPROM byte write operation can be executed when the mode selection bit, MODE, is cleared to zero. For byte write operations the desired EEPROM address should first be placed in the EEAH and EEAL registers, 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, indicating that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended. After the write operation is finished, the WREN bit will be cleared to zero

by hardware. Note that a byte erase operation will automatically be executed before a byte write operation is successfully activated.

Page Write Mode

Before a page write operation is executed, it is important to ensure that a relevant page erase operation has been successfully executed. The EEPROM page write operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page write. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM write enable control bit, namely WREN, is changed from "1" to "0", the internal page buffer will also be cleared. Note that when the WREN bit is changed from "0" to "1", the internal page buffer will not be cleared. A page write is initiated in the same way as a byte write initiation except that the EEPROM data can be written up to 16 bytes. The EEPROM address higher 7 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page write operation mode the lower 4-bit address value will automatically be incremented by one after each data byte is written into the EED register. However, the higher 7-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not "roll over". At this point any data write operations to the EED register will be invalid.

For page write operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers, 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, indicating that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended. After the write operation is finished, the WREN bit will be cleared to zero by hardware.

Write Protection

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

EEPROM Interrupt

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

Programming Considerations

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

When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. 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/write/erase operation is totally complete. Otherwise, the EEPROM read/write/erase operation will fail.

Programming Examples

Reading a Data Byte from the EEPROM – Polling Method

```
MOV  A, 40H                ; setup memory mointer lower byte MP1L
MOV  MP1L, A               ; MP1L points to EEC register
MOV  A, 01H                ; setup memory pointer high byte MP1H
MOV  MP1H, A
CLR  IAR1.4                ; clear MODE bit, select byte operatiom 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                ; set memory pointer low byte MP1L
MOV  MP1L, A               ; MP1L points to EEC register
MOV  A, 01H                ; set 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                ; set memory pointer low byte MP1L
MOV  MP1L, A               ; MP1L points to EEC register
MOV  A, 01H                ; set 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 write operations
SET  IAR1.5                ; start Write Cycle – set ER bit – executed immediately
                          ; after setting EREN bit
```

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

Writing a Data Byte to the EEPROM – Polling Method

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

Oscillators

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

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. 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 Oscillator	HIRC	4/8/12MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two oscillator sources, a high speed oscillator and a low speed oscillator. The high system clock is sourced from the internal 4/8/12MHz 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.
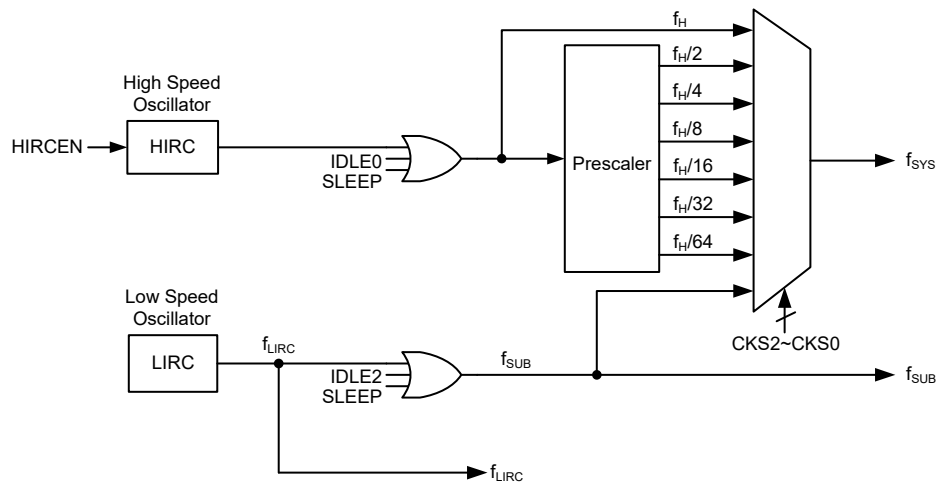
System Clock Configurations

Internal High Speed RC Oscillator – HIRC

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

Internal 32kHz Oscillator – LIRC

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

Operating Modes and System Clocks

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

System Clocks

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

The main system clock, can come from either a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.
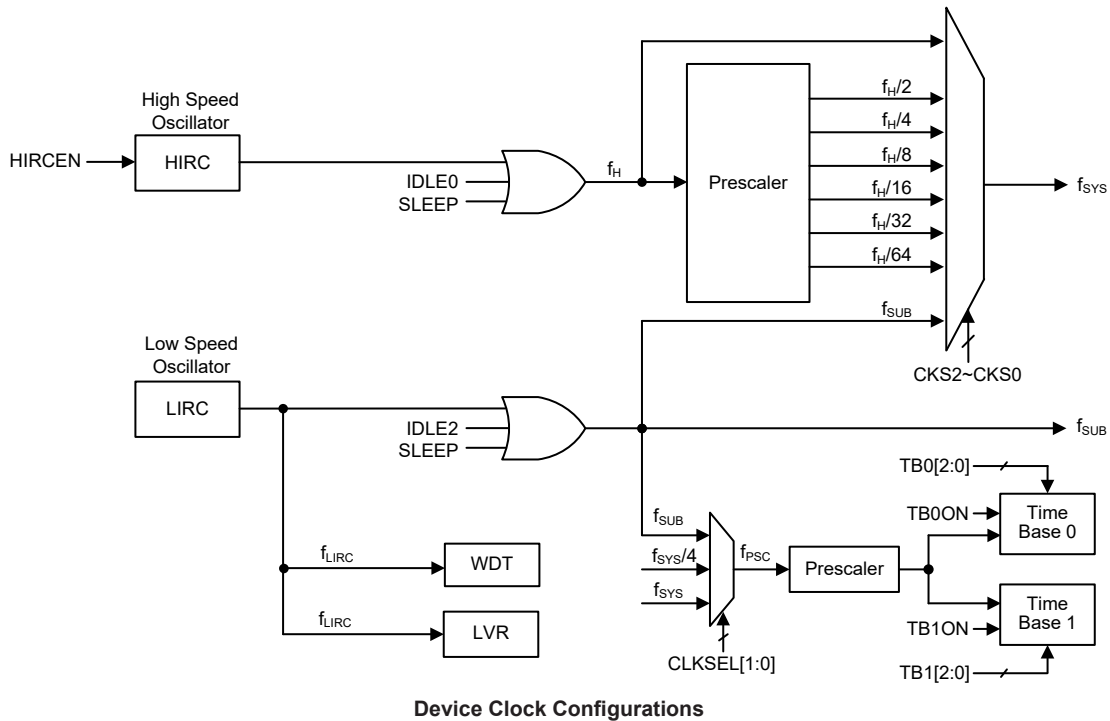
Device Clock Configurations

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

System Operation Modes

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

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

"x": don't care

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

2. The f_{LIRC} clock will be switched on or off which is controlled by the WDT function being enabled or disabled 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 the internal high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source which 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 an HALT instruction is executed and when the FHIDEN and FSIDEN bits in the SCC register are both low. In the SLEEP mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral function will also be stopped. However the f_{LIRC} clock will continue to operate if the WDT function is enabled.

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

The SCC and HIRCC registers are used to control the system clock and the HIRC oscillator configurations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	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 sytem clock period.

• **HIRCC Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection
00: 4MHz
01: 8MHz
10: 12MHz
11: 4MHz

When the HIRC oscillator is enabled or the HIRC frequency selection is changed by the application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1.

It is recommended that the HIRC frequency selected by these bits should be the same with the frequency determined by the configuration option to keep the HIRC frequency accuracy specified in the A.C. Characteristics.

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

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

Bit 0   **HIRCEN**: HIRC oscillator enable control
                    0: Disable
                    1: Enable

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.
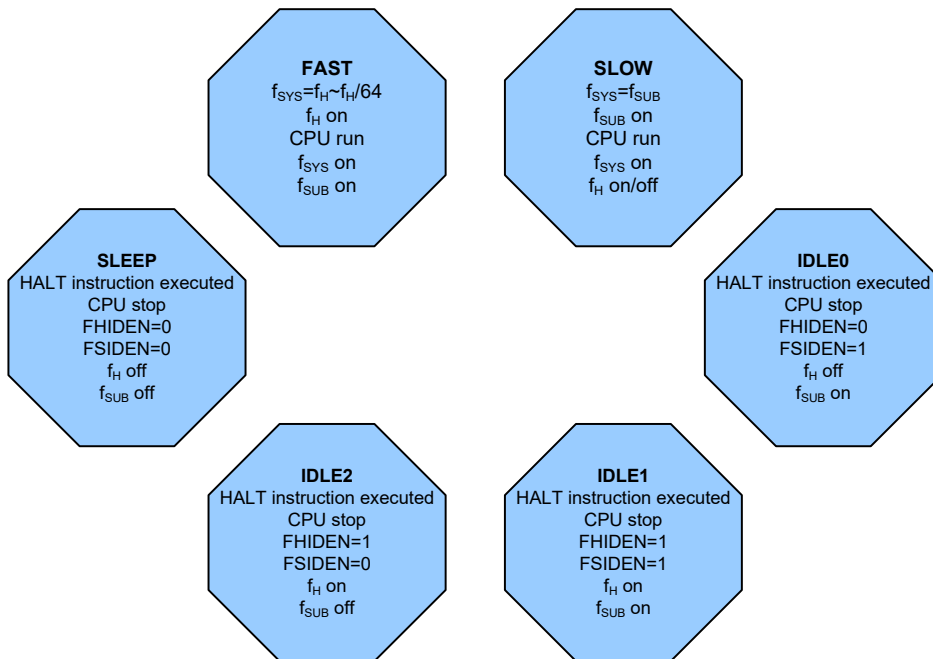
FAST Mode to SLOW Mode Switching

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.
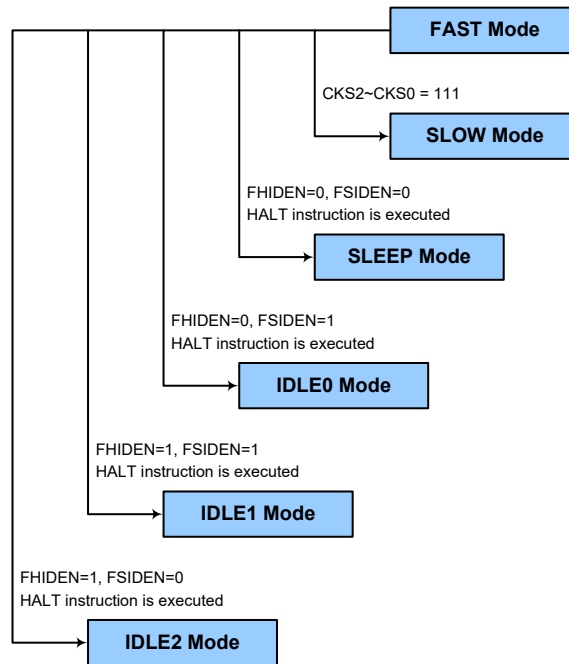
SLOW Mode to FAST Mode Switching

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~CKS0 bits should be set to "000"~"110" and then the system clock will respectively be switched to f_H~$f_H/64$.

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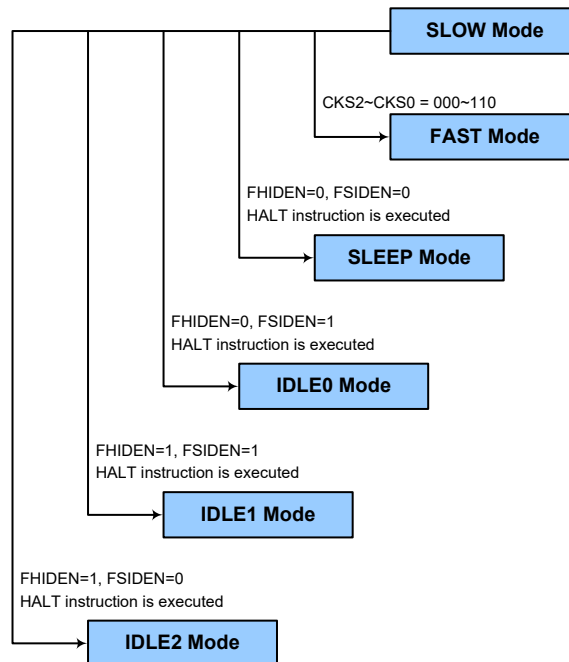
Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

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

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

Entering the IDLE2 Mode

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

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

Standby Current Considerations

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

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

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

Wake-up

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

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

• An external falling edge on Port A

• A system interrupt

• A WDT overflow

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

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

Watchdog Timer

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

Watchdog Timer Clock Source

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

Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period as well as Watchdog Timer the enable/disable and the MCU reset operation.

• **WDTC Register**

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

Bit 7~3   **WE4~WE0**: WDT function enable control
    10101: Disable
    01010: Enable
    Other values: Reset MCU

When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set high.

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

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

• **RSTFC Register**

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

"x": unknown

Bit 7~4   Unimplemented, read as "0"
Bit 3   **RSTF**: Reset control register software reset flag
    Refer to the Internal Reset Control section.
Bit 2   **LVRF**: LVR function reset flag
    Refer to the Low Voltage Reset section.

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

Bit 0   **WRF**: 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. This bit can only be cleared to zero by application program.

Watchdog Timer Operation

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

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

Watchdog Timer Function Control

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

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

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

Watchdog Timer
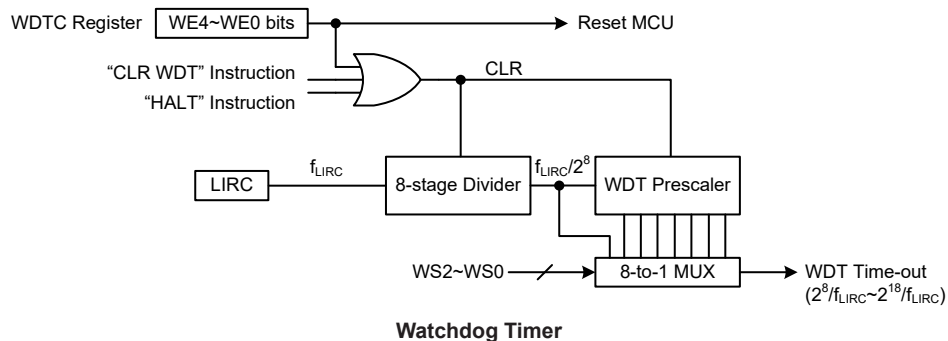
Reset and Initialisation

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

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

Reset Functions

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

Power-on Reset

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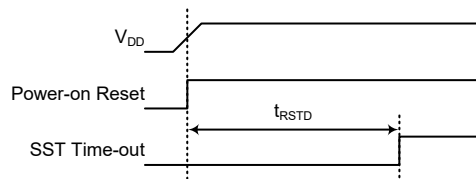
Power-On Reset Timing Chart

Internal Reset Control

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

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

Internal Reset Function Control

- **RSTC Register**

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

Bit 7~0   **RSTC7~RSTC0**: Reset function control
        01010101: No operation
        10101010: No operation
        Other values: Reset MCU

        If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET} and the RSTF bit in the RSTFC register will be set to 1. All resets will reset this register to POR value except the WDT time-out hardware warm reset.

- **RSTFC Register**

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

"x": unknown

Bit 7~4   Unimplemented, read as "0"

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

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

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

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

Low Voltage Reset – LVR

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

The LVR function is always enabled except in the SLEEP and IDLE mode with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of $0.9V$~V_{LVR} such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between $0.9V$~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual 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 are changed to any other values by environmental noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.
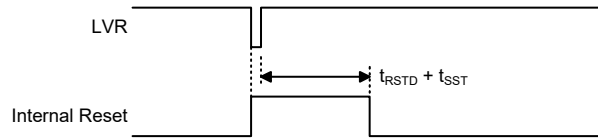
Low Voltage Reset Timing Chart

- **LVRC Register**

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

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

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

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

- **TLVRC Register**

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

Bit 7~2   Unimplemented, read as "0"

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

- **RSTFC Register**

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

"x": unknown

Bit 7~4   Unimplemented, read as "0"

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

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

Bit 1 **LRF**: LVRC register software reset flag

    0: Not occurred

    1: Occurred

This bit is set high if the LVRC register contains any non-defined LVR voltage register values. This in effect acts like a software reset function. This bit can only be cleared to zero by application program.

Bit 0 **WRF**: WDTC register software reset flag

Refer to the Watchdog Timer Control Register section.

IAP Reset

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

Watchdog Time-out Reset during Normal Operation

The Watchdog time-out Reset during normal operation is the same as the LVR Reset except that the Watchdog time-out flag TO will be set to "1".

WDT Time-out

t_{RSTD}

Internal Reset
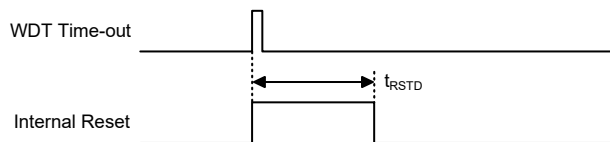
WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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

WDT Time-out

t_{SST}

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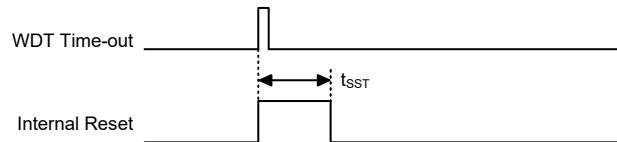
WDT Time-out Reset during SLEEP or IDLE Mode Timing Chart

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	-xxx xxxx	-uuu uuuu	-uuu uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
PBP	---- --00	---- --00	---- --uu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- 0x00	---- uuuu	---- uuuu
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- 0001	---- 0001	---- uuuu
STKPTR	0--- 0000	0--- 0000	u--- uuuu
IECC	0000 0000	0000 0000	uuuu uuuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
RSTC	0101 0101	0101 0101	uuuu uuuu
LVRC	0101 0101	0101 0101	uuuu uuuu
TLVRC	---- --01	---- --01	---- --uu
MFI0	0000 0000	0000 0000	uuuu uuuu
MFI1	-000 -000	-000 -000	-uuu -uuu
WDTC	0101 0011	0101 0011	uuuu uuuu
INTEG	0000 0000	0000 0000	uuuu uuuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu

Register	Power-on Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
INTC3	- - - 0 - - - 0	- - - 0 - - - 0	- - - u - - - u
PB	- - - - - - - 1	- - - - - - - 1	- - - - - - - u
PBC	- - - - - - - 1	- - - - - - - 1	- - - - - - - u
PBPU	- - - - - - - 0	- - - - - - - 0	- - - - - - - u
PCRL	0000 0000	0000 0000	uuuu uuuu
PCRH	-000 0000	-000 0000	-uuu uuuu
PSCR	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
TB0C	0 - - - - 0 0 0	0 - - - - 0 0 0	u - - - - u u u
TB1C	0 - - - - 0 0 0	0 - - - - 0 0 0	u - - - - u u u
PTMC0	0000 0 - - -	0000 0 - - -	uuuu u - - -
PTMC1	0000 0000	0000 0000	uuuu uuuu
PTMDL	0000 0000	0000 0000	uuuu uuuu
PTMDH	0000 0000	0000 0000	uuuu uuuu
PTMAL	0000 0000	0000 0000	uuuu uuuu
PTMAH	0000 0000	0000 0000	uuuu uuuu
PTMRPL	0000 0000	0000 0000	uuuu uuuu
PTMRPH	0000 0000	0000 0000	uuuu uuuu
EEAL	0000 0000	0000 0000	uuuu uuuu
EEAH	- - - - - 0 0 0	- - - - - 0 0 0	- - - - - u u u
EED	0000 0000	0000 0000	uuuu uuuu
PWRC	00- - 0000	00- - 0000	uu- - uuuu
IREFC	- - - 0 - - 0 0	- - - 0 - - 0 0	- - - u – u u
PVREF	0000 0000	0000 0000	uuuu uuuu
OPA1C	- - 0 - - - - -	- - 0 - - - - -	- - u - - - - -
OPA2C	- - 0 - - - - -	- - 0 - - - - -	- - u - - - - -
OPA3C	- - 0 - - - - -	- - 0 - - - - -	- - u - - - - -
GSC1	- - - - - 0 0 0	- - - - - 0 0 0	- - - - - u u u
AFEDA1C	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
AFEDA1L	0000 - - - -	0000 - - - -	uuuu - - - -
AFEDA1H	0000 0000	0000 0000	uuuu uuuu
AFEDA2C	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
AFEDA2L	0000 - - - -	0000 - - - -	uuuu - - - -
AFEDA2H	0000 0000	0000 0000	uuuu uuuu
AFEDA3C	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
AFEDA3L	0000 - - - -	0000 - - - -	uuuu - - - -
AFEDA3H	0000 0000	0000 0000	uuuu uuuu
AFEDA0C	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
AFEDA0L	0000 - - - -	0000 - - - -	uuuu - - - -
AFEDA0H	0000 0000	0000 0000	uuuu uuuu
MDUWR0	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR1	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR2	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR3	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR4	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR5	xxxx xxxx	0000 0000	uuuu uuuu
MDUWCTRL	00- - - - - -	00- - - - - -	uu- - - - - -
PGAC0	-00- 0000	-00- 0000	-uu- uuuu

Register	Power-on Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PGAC1	-000 --00	-000 --00	-uuu --uu
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	0-00 000-	0-00 000-	u-uu uuu-
ADCR1	000- -00-	000- -00-	uuu- -uu-
SINC3	1001 0011	1001 0011	uuuu uuuu
ADCR2	00-0 0010	00-0 0010	uu-u uuuu
ADCR3	0111 ---1	0111 ---1	uuuu ---u
ADCS	---0 0000	---0 0000	---u uuuu
CMPC	-000 0000	-000 0000	-uuu uuuu
PRFC	0--- 0000	0--- 0000	u--- 0000
TDAVRST	0000 0000	0000 0000	uuuu uuuu
TDAVRLEN	0000 0000	0000 0000	uuuu uuuu
TOPST	0000 0000	0000 0000	uuuu uuuu
TOPLEN	0000 0000	0000 0000	uuuu uuuu
TDAST	0000 0000	0000 0000	uuuu uuuu
TDALEN	0000 0000	0000 0000	uuuu uuuu
TCPST	0000 0000	0000 0000	uuuu uuuu
TCPLEN	0000 0000	0000 0000	uuuu uuuu
TCHS0	0000 0000	0000 0000	uuuu uuuu
TCHS1	0--- ----	0--- ----	u--- ----
SIMC0	111- 0000	111- 0000	uuu- uuuu
SIMC1	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2	0000 0000	0000 0000	uuuu uuuu
SIMTOC	0000 0000	0000 0000	uuuu uuuu
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
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
EEC	0000 0000	0000 0000	uuuu uuuu
FC0	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	uuuu uuuu

Register	Power-on Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
FC2	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
FARL	0000 0000	0000 0000	uuuu uuuu
FARH	-000 0000	-000 0000	-uuu uuuu
FD0L	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	uuuu uuuu
IFS	- - 0 0 0 0 0 -	- - 0 0 0 0 0 -	- - u u u u u -
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u

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. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A, [m]", where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	—	—	—	—	—	PB0
PBC	—	—	—	—	—	—	—	PBC0
PBPU	—	—	—	—	—	—	—	PBPU0

"—": Unimplemented, read as "0"

I/O Logic Function Registers List

Pull-high Resistors

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

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

• PxPU Register

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

PxPUn: I/O Port x Pin pull-high function control
　　　　0: Disable
　　　　1: Enable
　　　　The PxPUn bit is used to control the pin pull-high function. Here the "x" is the Port name which can be A or B. However, the actual available bits for each I/O Port may be different.

Port A Wake-up

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

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

• PAWU Register

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

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

I/O Port Control Registers

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

• **PxC Register**

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

PxCn: I/O Port x Pin type selection

    0: Output

    1: Input

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

Pin-shared Functions

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

Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The device includes Port "x" output function selection register "n", labeled as PxSn, and Input Function source pin 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 INTn, xTCKn, PTPI, 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
IFS	—	—	—	—	SDISDAPS1	SDISDAPS0	SCKSCLPS	—
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	—	—	—	—	—	—	PBS01	PBS00

Pin-shared Function Selection Register List

• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SDISDAPS1	SDISDAPS0	SCKSCLPS	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4   Unimplemented, read as "0"

Bit 3~2   **SDISDAPS1~SDISDSAPS0**: SDI/SDA input source pin selection
  00: PA7
  01: PA2
  10: Reserved
  11: PA7

Bit 1   **SCKSCLPS**: SCK/SCL input source pin selection
  0: PB0
  1: PA4

Bit 0   Unimplemented, read as "0"

• **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: PA3
  10: PA3
  11: AIN0

Bit 5~4   **PAS05~PAS04**: PA2 pin-shared function selection
  00: PA2
  01: SDI/SDA
  10: Reserved
  11: PA2

Bit 3~2   **PAS03~PAS02**: PA1 pin-shared function selection
  00: PA1
  01: PA1
  10: PA1
  11: AIN1

Bit 1~0   **PAS01~PAS00**: PA0 pin-shared function selection
  00: PA0
  01: SDO
  10: Reserved
  11: PA0

• **PAS1 Register**

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

Bit 7~6   **PAS17~PAS16**: PA7 pin-shared function selection
  00: PA7/INT2/PTPI
  01: SDI/SDA
  10: Reserved
  11: PA7/INT2/PTPI

Bit 5~4   **PAS15~PAS14**: PA6 pin-shared function selection
        00: PA6/CTCK1
        01: CTP1
        10: SDO
        11: Reserved

Bit 3~2   **PAS13~PAS12**: PA5 pin-shared function selection
        00: PA5/INT1/PTCK
        01: $\overline{SCS}$
        10: PA5/INT1/PTCK
        11: CMPO

Bit 1~0   **PAS11~PAS10**: PA4 pin-shared function selection
        00: PA4/INT0/CTCK0
   01: CTP0
        10: SCK/SCL
        11: PA4/INT0/CTCK0

• **PBS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PBS01	PBS00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2   Unimplemented, read as "0"

Bit 1~0   **PBS01~PBS00**: PB0 pin-shared function selection
        00: PB0/INT3
        01: SCK/SCL
        10: PTP
        11: CMPO

I/O Pin Structures

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

Logic Function Input/Output Structure
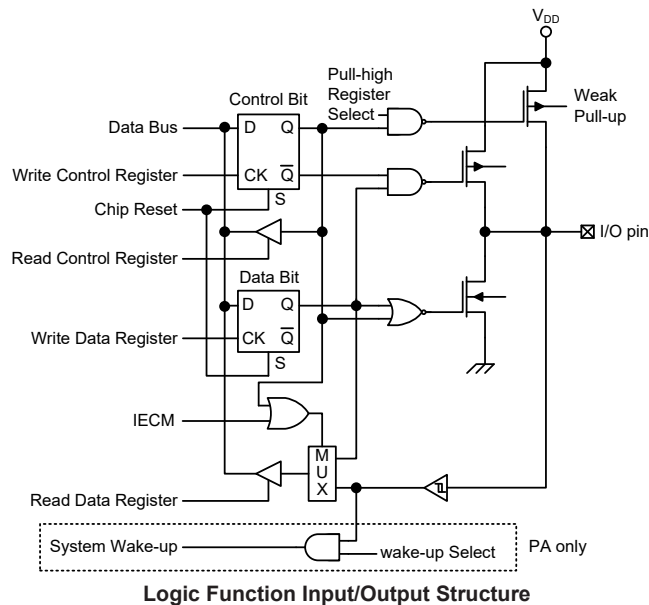
READ PORT Function

The READ PORT function is used to manage the reading of the output data from the data latch or I/O pin, which is specially designed for the IEC 60730 self-diagnostic test on the I/O function and A/D paths. There is a register, IECC, which is used to control the READ PORT function. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function. When a specific data pattern, "11001010", is written into the IECC register, the internal signal named IECM will be set high to enable the READ PORT function. If the READ PORT function is enabled, the value on the corresponding pins will be passed to the accumulator ACC when the read port instruction "mov acc, Px" is executed where the "x" stands for the corresponding I/O port name. Note that the READ PORT mode can only control the input path and will not affect the pin-shared function assignment and the current MCU operation. However, when the IECC register content is set to any other values rather than "11001010", the IECM internal signal will be cleared to 0 to disable the READ PORT function, and the reading path will be from the data latch. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function.

- **IECC Register**

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

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

READ PORT Function	Disabled		Enabled	
Port Control Register Bit – PxC.n	1	0	1	0
I/O function	Pin value	Data latch value	Pin value	
Digital input function				
Digital output function (except SIM)	0			
SIM: SCK/SCL, SDI/SDA	Pin value			
Analog function	0			

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

The additional function of the READ PORT mode is to check the A/D path. When the READ PORT function is disabled, the A/D path from the external pin to the internal analog input will be switched off if the A/D input pin function is not selected by the corresponding selection bits. For the MCU with A/D converter channels, such as A/D AIN0~AIN1, the desired A/D channel can be switched on by properly configuring the external analog input channel selection bits in the A/D Control Register together with the corresponding analog input pin function is selected. However, the additional function of the READ PORT mode is to force the A/D path to be switched on. For example, when the AIN0 is selected as the analog input channel as the READ PORT function is enabled, the AIN0 analog input path will be switched on even if the AIN0 analog input pin function is not selected. In this way, the AIN0 analog input path can be examined by internally connecting the digital output on this shared pin with the AIN0 analog input pin switch and then converting the corresponding digital data without any external analog input voltage connected.

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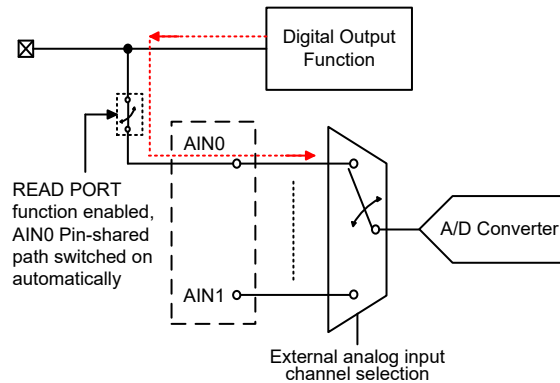
A/D Channel Input Path Internally Connection

Programming Considerations

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

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

Timer Modules – TM

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

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

Introduction

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

TM Function	CTM	PTM
Timer/Counter	√	√
Input capture	—	√
Compare match output	√	√
PWM output	√	√
Single pulse output	—	√
PWM alignment	Edge	Edge
PWM adjustment period & duty	Duty or Period	Duty or Period

TM Function Summary

TM Operation

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

TM Clock Source

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

TM Interrupts

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

TM External Pins

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

The PTM has another input pin, PTPI, which is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTIO1~PTIO0 bits in the PTMC1 register. There is another capture input, PTCK, for PTM capture input mode, which can be used as the external trigger input source except the PTPI pin.

The TMs each have one output pin with the label xTPn. When the TM is in the Compare Match Output Mode, this 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 external xTPn output pin is also the pin where the TM generates the PWM output waveform.

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

CTM0		CTM1		PTM	
Input	**Output**	**Input**	**Output**	**Input**	**Output**
CTCK0	CTP0	CTCK1	CTP1	PTCK, PTPI	PTP

TM External Pins
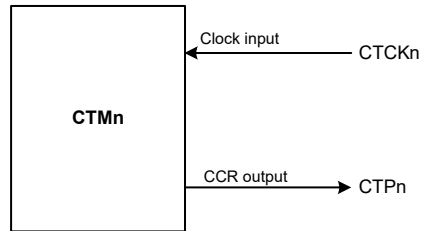
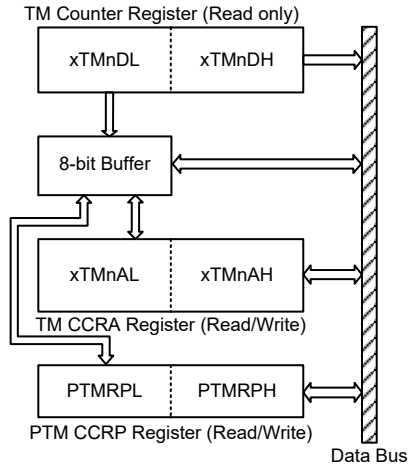**CTM Function Pin Block Diagram (n=0~1)**
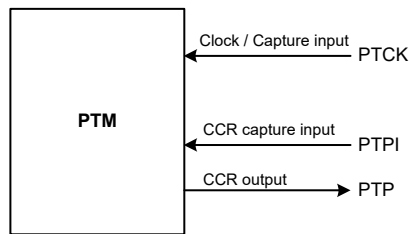
PTM Function Pin Block Diagram

Programming Considerations

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

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

TM Counter Register (Read only)

| xTMnDL | xTMnDH |

8-bit Buffer

| xTMnAL | xTMnAH |

TM CCRA Register (Read/Write)

| PTMRPL | PTMRPH |

PTM CCRP Register (Read/Write)

Data Bus

The following steps show the read and write procedures:

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

Compact Type TM – CTM

The Compact TM type contains three operating modes, which are Compare Match Output, Timer/Event Counter and PWM Output modes. The Compact TMs can also be controlled with an external input pin and can drive one external output pin.
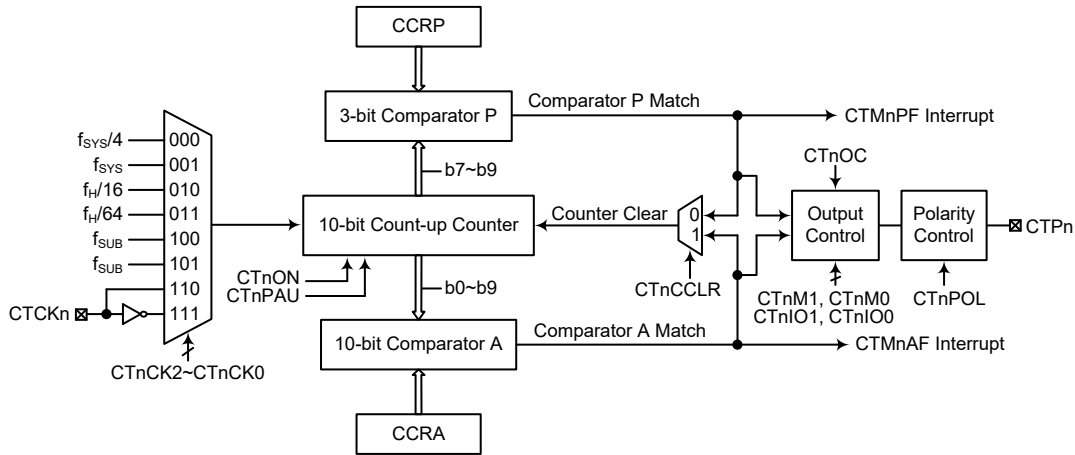
Note: As the CTMn external pins are pin-shared with other functions, so before using the CTMn function the relevant pin-shared function registers must be set properly to enable the CTMn pin function. The CTCKn pins, if used, must also be set as an input by setting the corresponding bits in the port control 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 is three bits wide whose value is compared with the highest 3 bits in the counter while the CCRA is the 10 bits and therefore compares with all counter bits.

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

Compact Type TM Register Description

Overall operation of the Compact TM is controlled using several registers. A read only register pair exists to store the internal counter 10-bit value, while a read/write register pair exists to store the internal 10-bit CCRA value. The remaining two registers are control registers which setup the different operating and control modes as well as the 3 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   **CTnPAU**: CTMn Counter Pause Control

0: Run
1: Pause

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

Bit 6~4   **CTnCK2~CTnCK0**: Select CTMn Counter clock

000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: CTCKn rising edge clock
111: CTCKn falling edge clock

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

Bit 3   **CTnON**: CTMn Counter On/Off Control

0: Off
1: On

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

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

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

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

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

- **CTMnC1 Register**

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

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

These bits setup the required operating mode for the 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**: Select CTMn external pin function

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

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

Timer/Counter Mode
  Unused

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

In the Compare Match Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a compare match occurs from the Comparator A. The CTMn output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the CTMn output pin should be setup using the CTnOC bit in the CTnMC1 register. Note that

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

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

Bit 3 **CTnOC**: CTPn Output control bit

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

PWM Output Mode
  0: Active low
  1: Active high

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

Bit 2 **CTnPOL**: CTPn Output polarity Control
  0: Non-invert
  1: Invert

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

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

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

Bit 0 **CTnCCLR**: Select CTMn Counter clear condition
  0: CTMn Comparatror P match
  1: CTMn Comparatror A match

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

• **CTMnDL Register**

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

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

• **CTMnDH Register**

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

Bit 7~2 Unimplemented, read as "0"

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

• **CTMnAL Register**

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

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

• **CTMnAH Register**

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

Bit 7~2 Unimplemented, read as "0"

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

Compact Type TM Operating Modes

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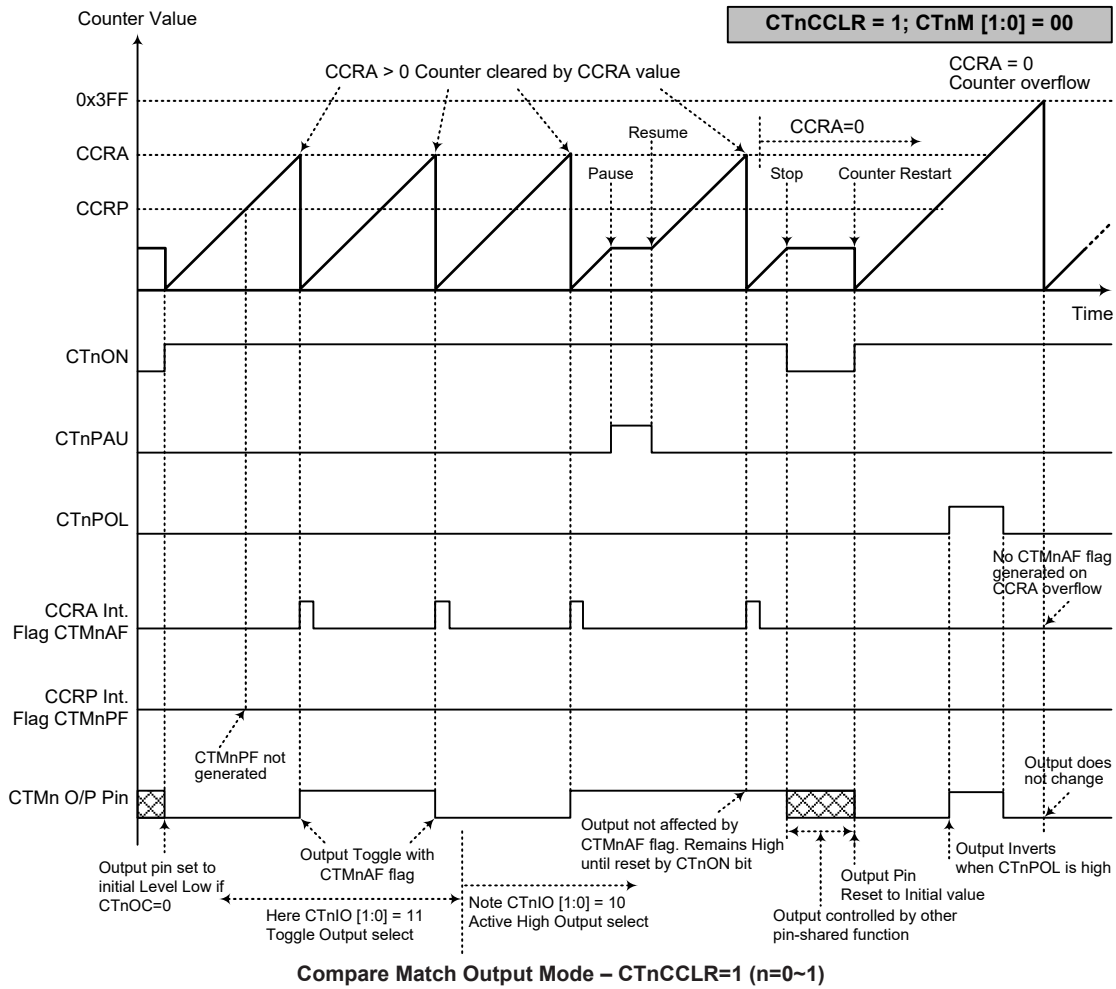
Compare Match Output Mode

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

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

As the name of the mode suggests, after a comparison is made, the CTMn output pin will change state. The CTMn output pin condition however only changes state when a CTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTMn output pin. The way in which the CTMn output pin changes state are determined by the condition of the CTnIO1 and CTnIO0 bits in the CTMnC1 register. The CTMn output pin can be selected using the CTnIO1 and CTnIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the CTMn output pin, which is setup after the CTnON bit changes from low to high, is setup using the CTnOC bit. Note that if the CTnIO1 and CTnIO0 bits are zero then no pin change will take place.
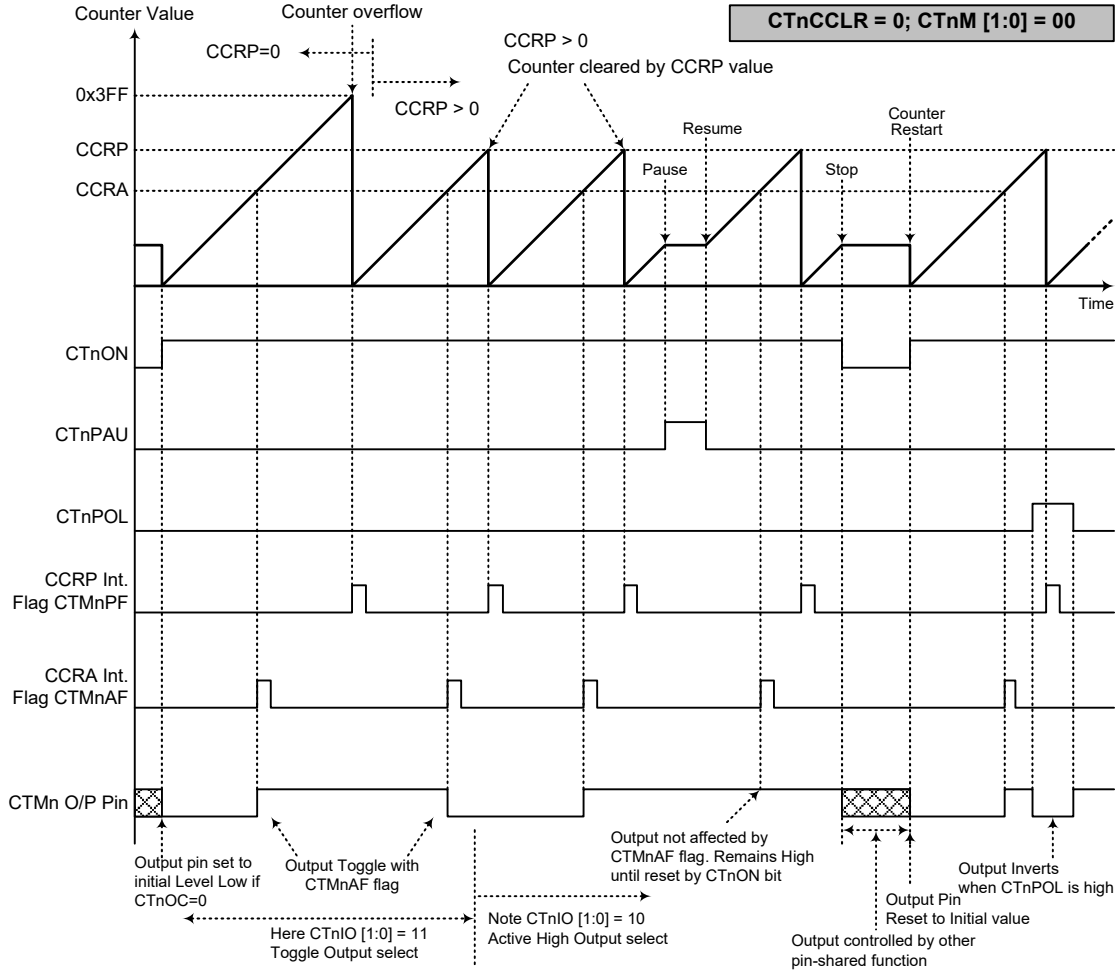
Compare Match Output Mode – CTnCCLR=0 (n=0~1)

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

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

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

2. The CTMn output pin controlled only by the CTMnAF flag

3. The output pin reset to initial state by a CTnON rising edge

4. The CTMnPF flags is not generated when CTnCCLR=1

Timer/Counter Mode

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

PWM Output Mode

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

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

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

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

If f_{SYS}=8MHz, CTMn clock source is f_{SYS}/4, CCRP=100b, CCRA=128,

The CTMn PWM output frequency=(f_{SYS}/4)/512=f_{SYS}/2048=3.9063kHz, duty=128/512=25%.

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

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

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

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

PWM Output Mode – CTnDPX=0 (n=0~1)
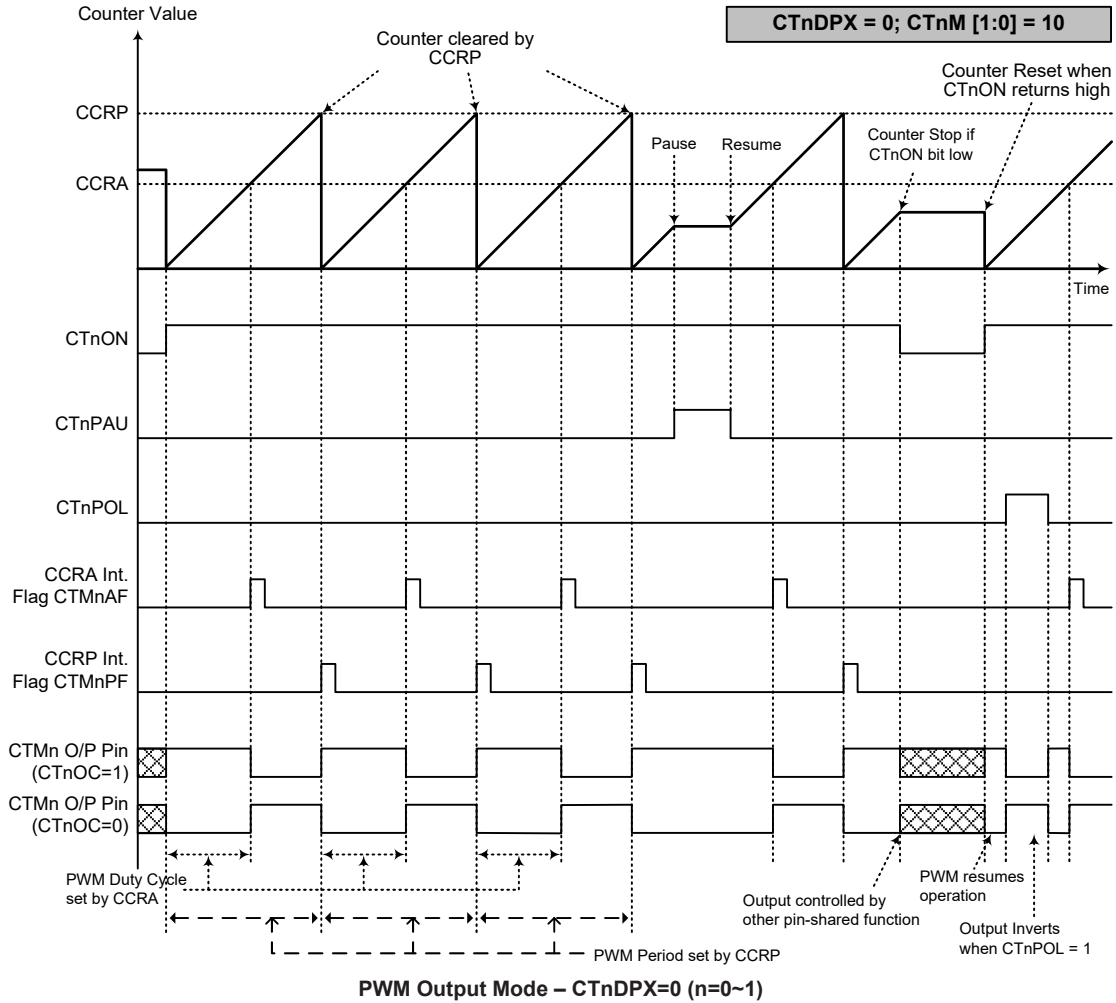
Note: 1. Here CTnDPX=0 – Counter cleared by CCRP

2. A counter clear sets PWM Period

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

4. The CTnCCLR bit has no influence on PWM operation

PWM Output Mode – CTnDPX=1 (n=0~1)
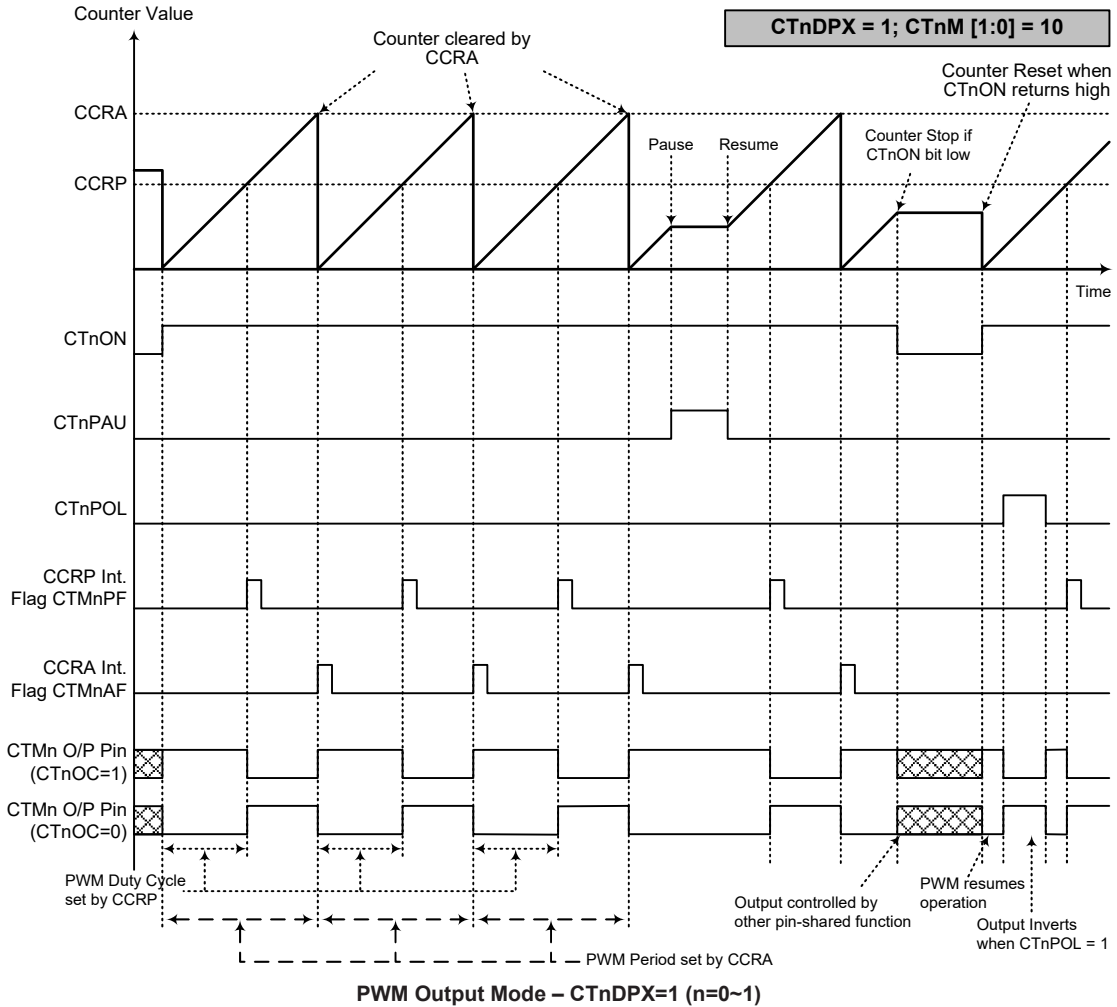
Note: 1. Here CTnDPX=1 – Counter cleared by CCRA

2. A counter clear sets PWM Period

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

4. The CTnCCLR bit has no influence on PWM operation

Periodic Type TM – PTM

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

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

16-bit Periodic Type TM Block Diagram

Periodic TM Operation

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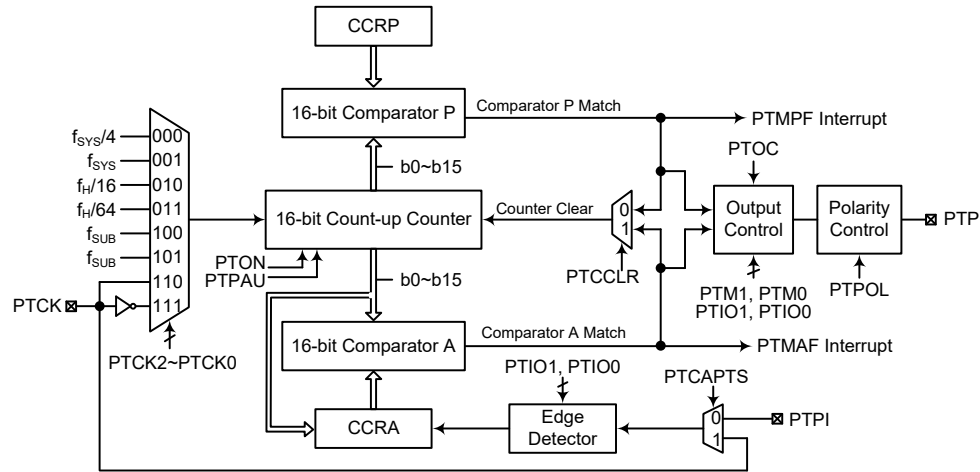 or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP and CCRA comparators are 16-bit wide whose value is respectively compared with all counter bits.

The only way of changing the value of the 16-bit counter using the application program is to clear the counter by changing the PTON 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 PTM interrupt signal will also usually be generated. The Periodic Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control an output pin. All operating setup conditions are selected using relevant internal registers.

Periodic Type TM Register Description

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

| Register | Bit | | | | | | | |
Name	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	D15	D14	D13	D12	D11	D10	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	D15	D14	D13	D12	D11	D10	D9	D8
PTMRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMRPH	D15	D14	D13	D12	D11	D10	D9	D8

16-bit Periodic TM Register List

- **PTMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7   **PTPAU**: PTM 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 PTM 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   **PTCK2~PTCK0**: Select PTM Counter clock

     000: $f_{SYS}/4$

     001: f_{SYS}

     010: $f_H/16$

     011: $f_H/64$

     100: f_{SUB}

     101: f_{SUB}

     110: PTCK rising edge clock

     111: PTCK falling edge clock

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

Bit 3   **PTON**: PTM Counter On/Off control

     0: Off

     1: On

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

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

Bit 2~0   Unimplemented, read as "0"

• **PTMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
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   **PTM1~PTM0**: Select PTM Operating Mode
     00: Compare Match Output Mode
     01: Capture Input Mode
     10: PWM Output Mode or Single Pulse Output Mode
     11: Timer/Counter Mode

These bits setup the required operating mode for the PTM. To ensure reliable operation the PTM should be switched off before any changes are made to the PTM1 and PTM0 bits. In the Timer/Counter Mode, the PTM output pin state is undefined.

Bit 5~4   **PTIO1~PTIO0**: Select PTM external pin function selection
Compare Match Output Mode
     00: No change
     01: Output low
     10: Output high
     11: Toggle output
PWM Output Mode/Single Pulse Output Mode
     00: PWM output inactive state
     01: PWM output active state
     10: PWM output
     11: Single Pulse Output
Capture Input Mode
     00: Input capture at rising edge of PTPI or PTCK
     01: Input capture at falling edge of PTPI or PTCK
     10: Input capture at rising/falling edge of PTPI or PTCK
     11: Input capture disabled

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

In the Compare Match Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a compare match occurs from Comparator A. The PTM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the PTM output pin should be setup using the PTOC bit in the PTMC1 register. Note that the output level requested by the PTIO1 and PTIO0 bits must be different from the initial value setup using the PTOC bit otherwise no change will occur on the PTM output pin when a compare match occurs. After the PTM output pin changes state, it can be reset to its initial level by changing the level of the PTON bit from low to high.

In the PWM Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a certain compare match condition occurs. The PTM output function is modified by changing these two bits. It is necessary to only change the values of the PTIO1 and PTIO0 bits only after the PTM has been switched off. Unpredictable PWM outputs will occur if the PTIO1 and PTIO0 bits are changed when the PTM is running.

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

Bit 2   **PTPOL**: PTM PTP Output polarity control
 0: Non-invert
 1: Invert

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

Bit 1   **PTCAPTS**: PTM Capture Trigger Source selection
 0: From PTPI pin
 1: From PTCK pin

Bit 0   **PTCCLR**: PTM Counter Clear condition selection
 0: Comparator P match
 1: Comparator A match

This bit is used to select the method which clears the counter. Remember that the Periodic TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the PTCCLR 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 PTCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

• **PTMDL 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**: PTM Counter Low Byte Register bit 7 ~ bit 0
 PTM 16-bit Counter bit 7 ~ bit 0

• **PTMDH Register**

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

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

• **PTMAL 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**: PTM CCRA Low Byte Register bit 7 ~ bit 0
　　　　　　PTM 16-bit CCRA bit 7 ~ bit 0

• **PTMAH 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**: PTM CCRA High Byte Register bit 7 ~ bit 0
　　　　　　PTM 16-bit CCRA bit 15 ~ bit 8

• **PTMRPL 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**: PTM CCRP Low Byte Register bit 7 ~ bit 0
　　　　　　PTM 16-bit CCRP bit 7 ~ bit 0

• **PTMRPH 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**: PTM CCRP High Byte Register bit 7 ~ bit 0
　　　　　　PTM 16-bit CCRP bit 15 ~ bit 8

Periodic Type TM Operation Modes

The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTM1 and PTM0 bits in the PTMC1 register.
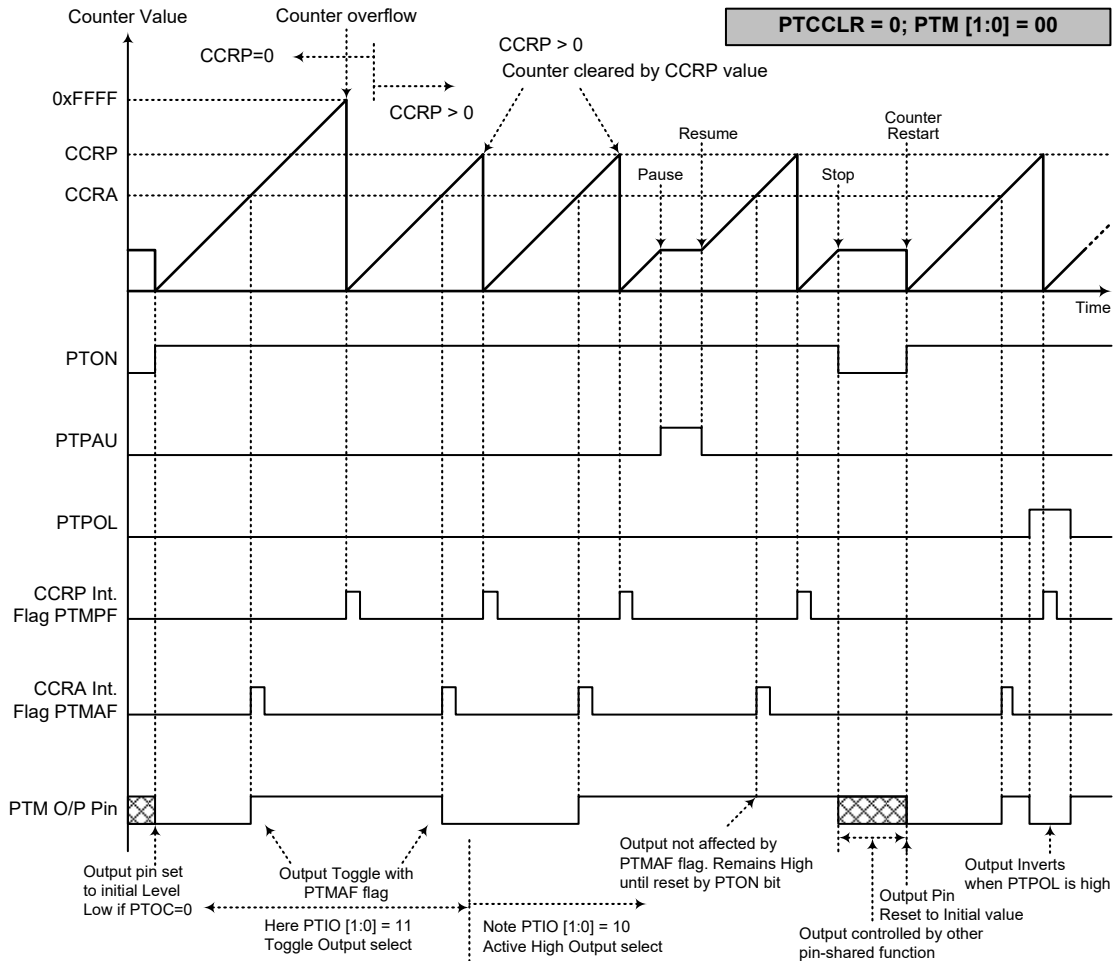
Compare Match Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 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 PTCCLR 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 PTMAF and PTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the PTCCLR bit in the PTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTCCLR is high no PTMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be cleared to "0".

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

As the name of the mode suggests, after a comparison is made, the PTM output pin will change state. The PTM output pin condition however only changes state when a PTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the PTM output pin. The way in which the PTM output pin changes state are determined by the condition of the PTIO1 and PTIO0 bits in the PTMC1 register. The PTM output pin can be selected using the PTIO1 and PTIO0 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 PTM output pin, which is setup after the PTON bit changes from low to high, is setup using the PTOC bit. Note that if the PTIO1 and PTIO0 bits are zero then no pin change will take place.

Compare Match Output Mode – PTCCLR=0

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

2. The PTM output pin is controlled only by the PTMAF flag

3. The output pin is reset to its initial state by a PTON bit rising edge
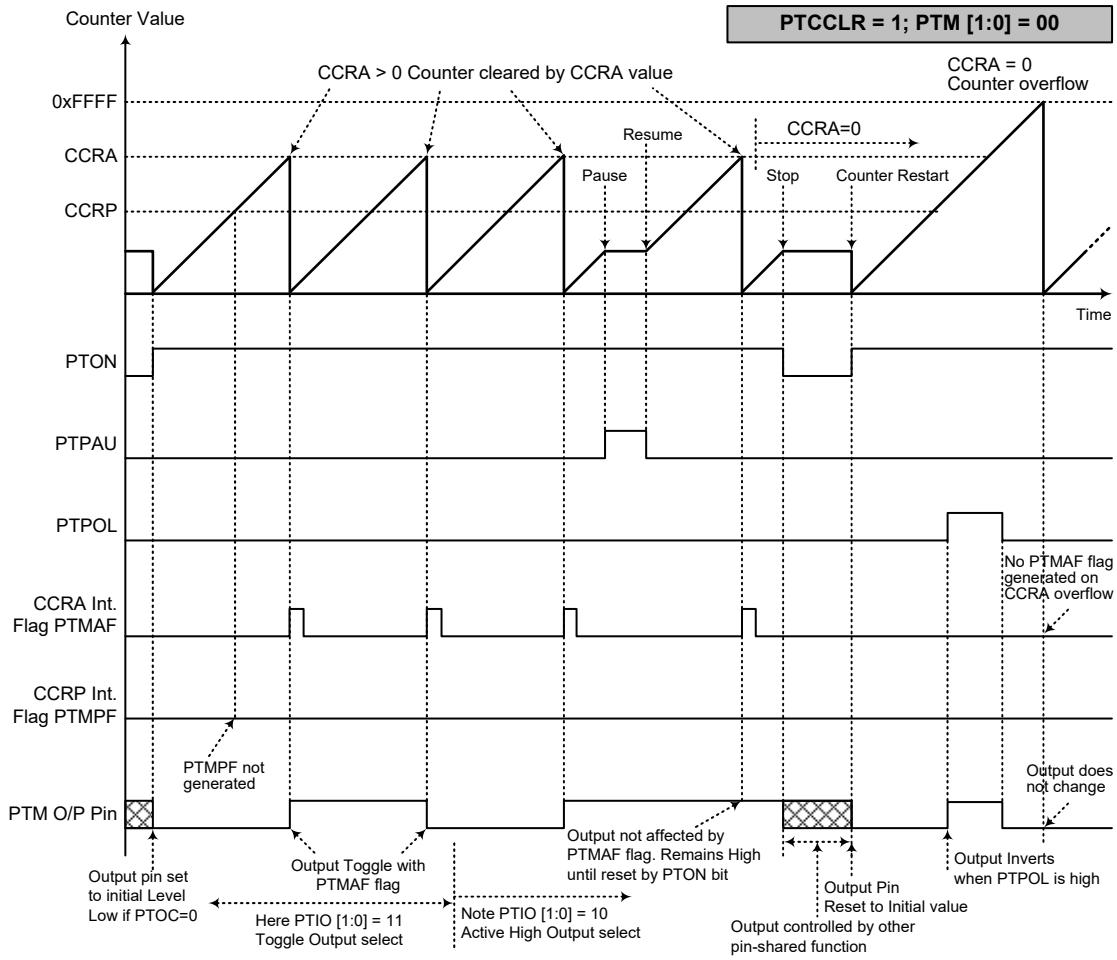
Compare Match Output Mode – PTCCLR=1

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

2. The PTM output pin is controlled only by the PTMAF flag

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

4. A PTMPF flag is not generated when PTCCLR=1

Timer/Counter Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 11. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the PTM 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 PTM 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 PTM1 and PTM0 in the PTMC1 register should be set to 10. The PWM function within the PTM 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 PTM 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 PTCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA registers are used to generate the PWM waveform, the CCRP is used to clear the internal counter and thus control the PWM waveform frequency, while the CCRA is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The PTOC bit in the PTMC1 register is used to select the required polarity of the PWM waveform while the two PTIO1 and PTIO0 bits are used to enable the PWM output or to force the PTM output pin to a fixed high or low level. The PTPOL bit is used to reverse the polarity of the PWM output waveform.

- **16-bit PTM, PWM Output Mode, Edge-aligned Mode**

CCRP	1~65535	0
Period	1~65535	65536
Duty	CCRA	

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

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

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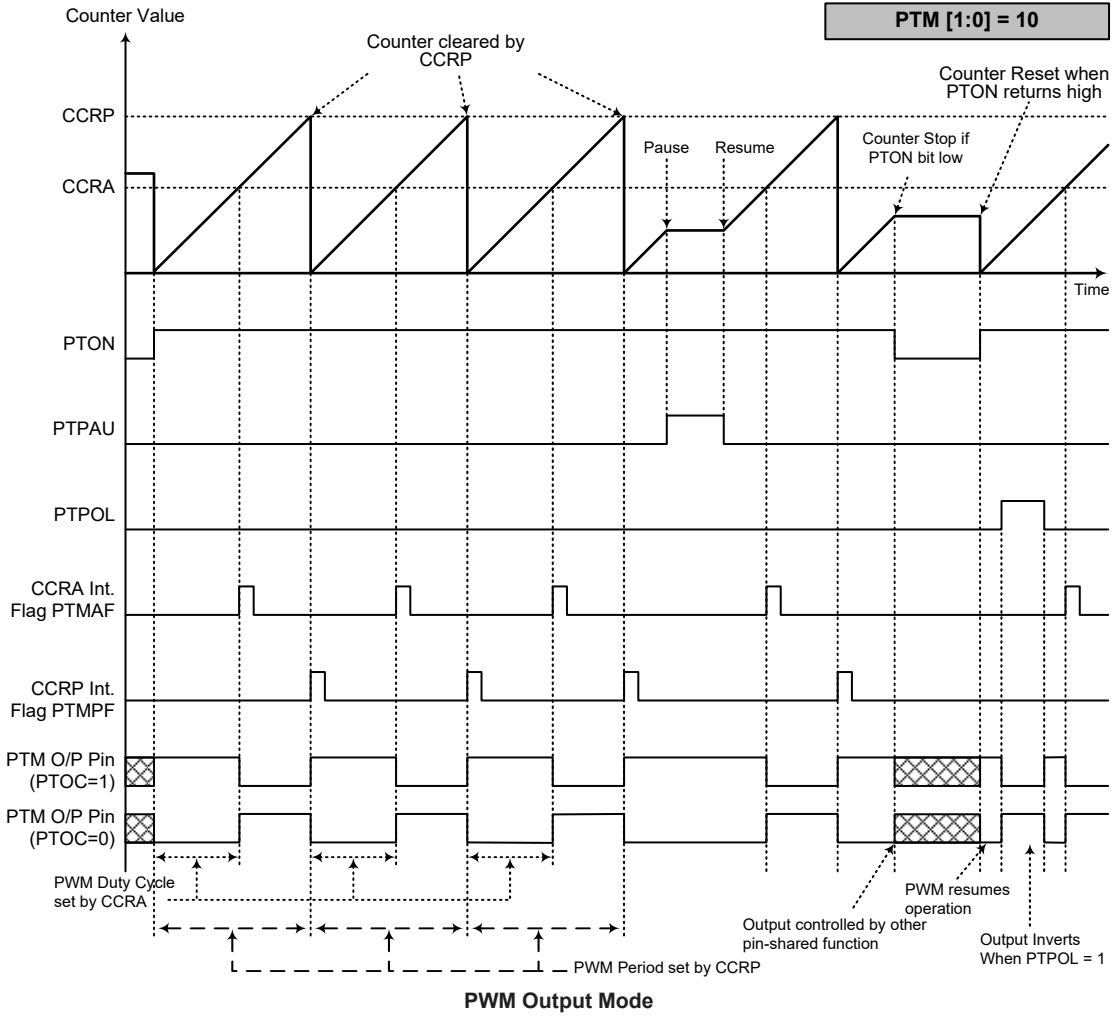
PWM Output Mode

Note: 1. The counter is cleared by CCRP

    2. A counter clear sets the PWM Period

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

    4. The PTCCLR bit has no influence on PWM operation

Single Pulse Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 and also the PTIO1 and PTIO0 bits should be set to 11. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

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

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

Single Pulse Generation
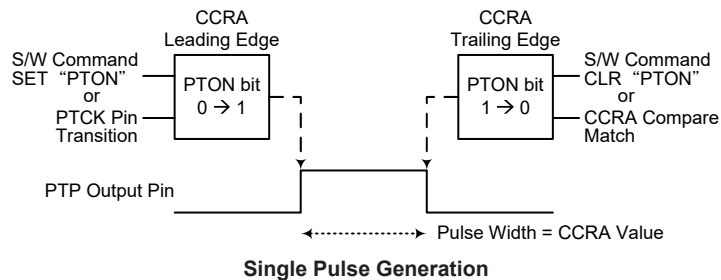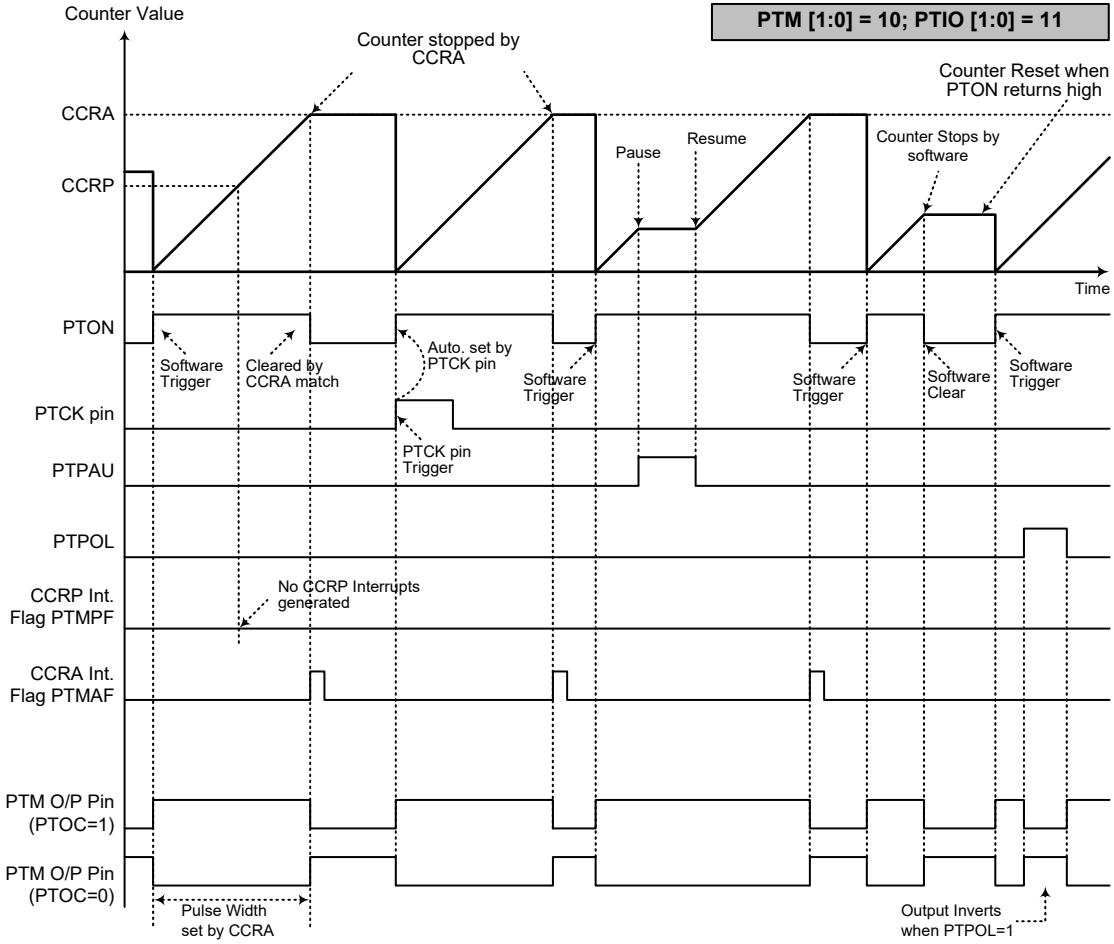
Single Pulse Output Mode

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

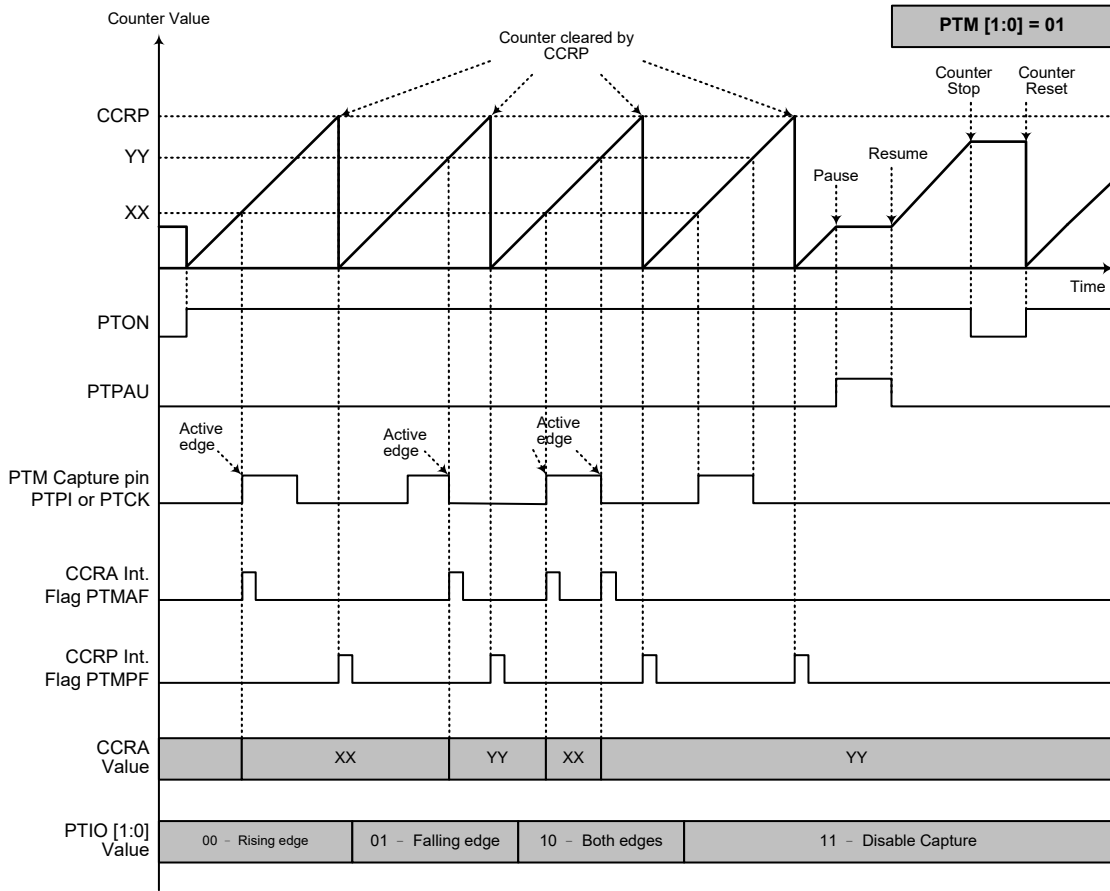**Capture Input Mode**

To select this mode bits PTM1 and PTM0 in the PTMC1 register should be set to 01. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPI or PTCK pin, selected by the PTCAPTS bit in the PTMC1 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTIO1 and PTIO0 bits in the PTMC1 register. The counter is started when the PTON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the PTPI or PTCK pin the present value in the counter will be latched into the CCRA registers and a PTM interrupt generated. Irrespective of what events occur on the PTPI or PTCK pin, the counter will continue to free run until the PTON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTIO1 and PTIO0 bits can select the active trigger edge on the PTPI or PTCK pin to be a rising edge, falling edge or both edge types. If the PTIO1 and PTIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPI or PTCK pin, however it must be noted that the counter will continue to run. The PTCCLR, PTOC and PTPOL bits are not used in this mode.

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

Capture Input Mode

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

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

3. PTCCLR bit not used

4. No output function – PTOC and PTPOL bits are not used

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

6. The capture input mode cannot be used if the selected PTM counter clock is not available

Internal Reference Voltage Generator

The device includes an internal reference voltage generator to provide an accurate reference voltage V_{IREF}. This reference voltage can be used as the internal 12-bit D/A Converter or 24-bit Delta Sigma A/D Converter reference voltage or can be output through the DACVREF pin by properly settings to use for other circuits. Refer to the Internal Reference Voltage Characteristics section for more information.
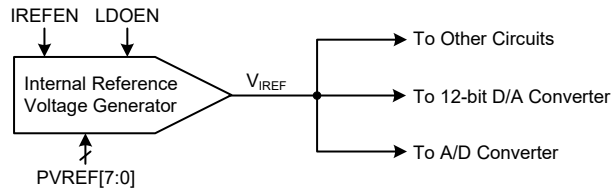
Note: The internal reference voltage generator is controlled by the IREFEN and LDOEN bits.

IREFEN	LDOEN	Internal Reference Voltage Generator
0	0	Off
0	1	On
1	0	On
1	1	On

Internal Reference Voltage Register Description

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

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

Internal Reference Voltage Register List

- **IREFC Register**

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

Bit 7~5   Unimplemented, read as "0"

Bit 4   **IREFEN**: Internal Reference Voltage Generator control
       0: Disable
       1: Enable

     This bit controls the internal reference voltage generator on/off. When this bit is set high, the internal reference voltage V_{IREF} can be generated and used as the reference voltage. If the internal referenc voltage is not used by any circuits, the IREFEN bit should be cleared to zero to reduce power consumption.

Bit 3~2   Unimplemented, read as "0"

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

• **PVREF Register**

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

Bit 7~0   **D7~D0**: Internal Reference Voltage Generator fine tune control

Setting the register can fine tune the internal reference voltage with a range of -60mV ~ +60mV (based on PVREF=80H). When the PVREF register value is increased by one, the internal reference voltage will decrease around 500μV, vice verse.

Measurement Circuit

This device includes a measure circuit which is composed of four 12-bit D/A Converters, three operational amplifiers and a comparator. Each D/A converter can provide a certain voltage from 0 to $V_{DACVREF}$ to the positive input of the internal operational amplifiers and the D/A converter output voltage can be connected to the A/D converter input for measurement. By proper setup, the OPA output voltage can be connected to the A/D converter internal input channel for measurement. A comparator interrupt request will take place when the comparator interrupt request flag, CMPF, is set, which occurs when the specified OPA input signal is detected.
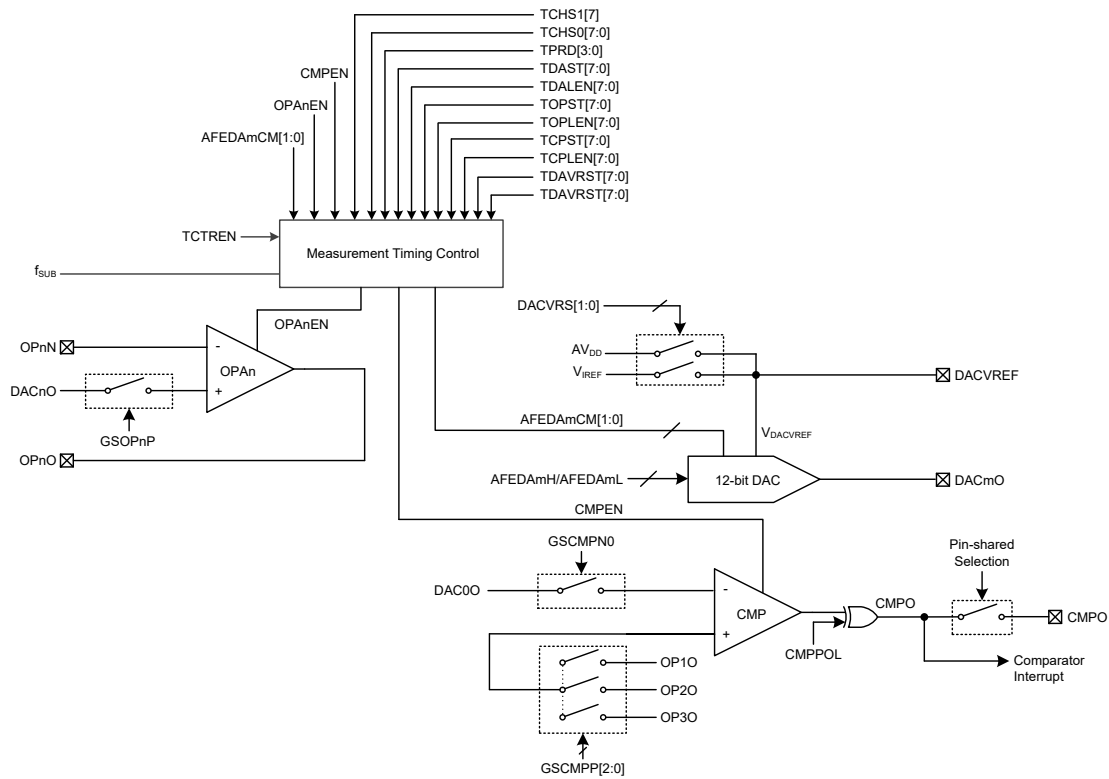
Measurement Circuit Structure (m=0~3, n=1~3)

Measurement Circuit Register Description

The measurement circuit is controlled by a series registers. The PRFC, TOPST, TDAST, TCPST, TDAVRST, TOPLEN, TDALEN, TCPLEN, TDAVRLEN, TCHS0 and TCHS1 registers are used for timing control. The IREFC, AFEDAmC, AFEDAmL, AFEDAmH registers are used for the DAC

control. The OPAnC and GSC1 registers are used for the OPA control. The CMPC register is used for CMP control.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PRFC	TCTREN	—	—	—	TPRD3	TPRD2	TPRD1	TPRD0
TOPST	D7	D6	D5	D4	D3	D2	D1	D0
TOPLEN	D7	D6	D5	D4	D3	D2	D1	D0
TDAVRST	D7	D6	D5	D4	D3	D2	D1	D0
TDAVRLEN	D7	D6	D5	D4	D3	D2	D1	D0
TDAST	D7	D6	D5	D4	D3	D2	D1	D0
TDALEN	D7	D6	D5	D4	D3	D2	D1	D0
TCPST	D7	D6	D5	D4	D3	D2	D1	D0
TCPLEN	D7	D6	D5	D4	D3	D2	D1	D0
TCHS0	TCPEN	TDA3EN	TDA2EN	TDA1EN	TDA0EN	TOP3EN	TOP2EN	TOP1EN
TCHS1	TDAVREN	—	—	—	—	—	—	—
IREFC	—	—	—	IREFEN	—	—	DACVRS1	DACVRS0
AFEDAmC	—	—	—	—	—	—	AFEDAmCM1	AFEDAmCM0
AFEDAmL	D3	D2	D1	D0	—	—	—	—
AFEDAmH	D11	D10	D9	D8	D7	D6	D5	D4
OPAnC	—	—	OPAnEN	—	—	—	—	—
GSC1	—	—	—	—	—	GSOP3P	GSOP2P	GSOP1P
CMPC	—	CMPPOL	CMPO	CMPEN	GSCMPN0	GSCMPP2	GSCMPP1	GSCMPP0

Measurement Circuit Register List (m=0~3, n=1~3)

• **PRFC Register**

Bit	7	6	5	4	3	2	1	0
Name	TCTREN	—	—	—	TPRD3	TPRD2	TPRD1	TPRD0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7   **TCTREN**: Timing control enable/disable
             0: Disable
             1: Enable

Bit 6~4   Unimplemented, read as "0"

Bit 3~0   **TPRD3~TPRD0**: Timing control clock period selection ($f_{PRF}=f_{SUB}/16384$)
             0000: f_{PRF}
             0001: $2/f_{PRF}$
             0010: $3/f_{PRF}$
             0011: $4/f_{PRF}$
             0100: $5/f_{PRF}$
             0101: $6/f_{PRF}$
             0110: $7/f_{PRF}$
             0111: $8/f_{PRF}$
             1000: $9/f_{PRF}$
             1001: $10/f_{PRF}$
             1010: $11/f_{PRF}$
             1011: $12/f_{PRF}$
             1100: $13/f_{PRF}$
             1101: $14/f_{PRF}$
             1110: $15/f_{PRF}$
             1111: $16/f_{PRF}$

- **TOPST 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**: OPA start delay time setting

Note: This register can define up to 255 ticks. $f_{base}=f_{SUB}/128$.

When the TCTREN bit is high, the timing control circuit can delay several ticks according to the TOPST definition before starting the OPA. For example, each tick time is $(1/f_{base})$, TOPST=4, timing control will delay a duration of $(1/f_{base}) \times 4$ then start the OPA.

- **TOPLEN 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**: OPA turned on time duration setting

Note: This register can define up to 255 ticks. $f_{base}=f_{SUB}/128$.

When the TCTREN bit is high, the timing control circuit will start the OPA and keep it on for several ticks defined by the TOPLEN register, then turn off the OPA. For example, each tick time is $(1/f_{base})$, TOLEN=50, timing control will start the OPA and last for a duration of $(1/f_{base}) \times 50$ then turn off the OPA.

- **TDAVRST 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**: DACVREF start delay time setting

Note: Same as the TOPST register, the TDAST register is used to control the DACVREF start delay time.

- **TDAVRLEN 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**: DACVREF turned on time duration setting

Note: Same as the TOPLEN register, the TDALEN register is used to control the DACVERF turned on time duration.

- **TDAST 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**: DAC start delay time setting

Note: Same as the TOPST register, the TDAST register is used to control the DAC start delay time.

• **TDALEN 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**: DAC turned on time duration setting

Note: Same as the TOPLEN register, the TDALEN register is used to control the DAC turned on time duration.

• **TCPST 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**: Comparator start delay time setting

Note: Same as the TOPST register, the TCPST register is used to comtrol the comparator start delay time.

• **TCPLEN 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**: Comparator turned on time duration setting

Note: Same as the TOPLEN register, the TCPLEN register is used to control the comparator turn on time duration.

• **TCHS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	TCPEN	TDA3EN	TDA2EN	TDA1EN	TDA0EN	TOP3EN	TOP2EN	TOP1EN
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   **TCPEN**: Timing control enable for comparator
            0: Disable
            1: Enable

Bit 6   **TDA3EN**: Timing control enable for DAC3
            0: Disable
            1: Enable

Bit 5   **TDA2EN**: Timing control enable for DAC2
            0: Disable
            1: Enable

Bit 4   **TDA1EN**: Timing control enable for DAC1
            0: Disable
            1: Enable

Bit 3   **TDA0EN**: Timing control enable for DAC0
            0: Disable
            1: Enable

Bit 2   **TOP3EN**: Timing control enable for OPA3
            0: Disable
            1: Enable

Bit 1   **TOP2EN**: Timing control enable for OPA2
               0: Disable
               1: Enable

Bit 0   **TOP1EN**: Timing control enable for OPA1
               0: Disable
               1: Enable

Note: 1. When the TCTREN bit is low, this register is disabled. The DAC, OPA and CMP functions are enabled by the AFEDAmCM[1:0], OPAnEN and CMPEN bit respectively (m=0~3, n=1~3).

    2. When the TCTREN bit is high, the TCHS0 register has higher control priority, which is higher than AFEDAmCMx, OPAnEN and CMPEN bits for DAC, OPA and CMP.

    For example (TCTREN=1):

    TOP1EN=1, OPA1EN=0 or 1, the OPA1 is periodically controlled (on or off) by timing control.

    TOP1EN=0, the OPA1 will be set to constant on or off controlled by the OPA1EN bit.

    TDAmEN (m=0~3)=0, the corresponding DAC is disabled, so the corresponding DAC output is in a ground state.

- **TCHS1 Register**

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

Bit 7   **TDAVREN**: Timing control enable for DACVREF
               0: Disable
               1: Enable

Bit 6~0   Unimplemented, read as "0"

Note: 1. When the TCTREN bit is low, this register is disabled. The DACVREF pin is enabled by the IREFEN bit.

    2. When the TCTREN bit is high, the TCHS1 register has higher control priority, which is higher than the IREFEN bit for DACVREF pin.

- **IREFC Register**

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

Bit 7~5   Unimplemented, read as "0"

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

Bit 3~2   Unimplemented, read as "0"

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

• **AFEDAmC Register (m=0~3)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	AFEDAmCM1	AFEDAmCM0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2   Unimplemented, read as "0"

Bit 1~0   **AFEDAmCM1~AFEDAmCM0**: 12-bit D/A Converter mode control
       00: DAC Disable, output in a high impedance state
       01/11: DAC Enable
       10: DAC Disable, output in a ground state

• **AFEDAmH & AFEDAmL Registers (m=0~3)**

Register	AFEDAmH								AFEDAmL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	—	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	0	0	0	0	0	0	0	0	—	—	—	—

"—": Unimplemented, read as "0"

D11~D0: 12-bit D/A Converter output control bits
       The bit 7 ~ bit 0 in the AFEDAmH register combine with the bit 7 ~ bit 4 in the AFEDAmL register to form a 12-bit DAC value of 0~4095.
       DAC Output Voltage $(V_{DACO}) = V_{DACVREF} \times (DAC\ value/4096)$

Note: It is necessary to firstly write into the AFEDAmL register followed by the AFEDAmH register.

• **OPAnC Register (n=1~3)**

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

Bit 7~6   Unimplemented, read as "0"

Bit 5   **OPAnEN**: OPAn enable/disable control
       0: Disable
       1: Enable

Bit 4~0   Unimplemented, read as "0"

• **GSC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	GSOP3P	GSOP2P	GSOP1P
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3   Unimplemented, read as "0"

Bit 2   **GSOP3P**: OP3P Selection switch for DAC3O
       0: Off
       1: On

Bit 1   **GSOP2P**: OP2P Selection switch for DAC2O
       0: Off
       1: On

Bit 0 **GSOP1P**: OP1P Selection switch for DAC1O
    0: Off
    1: On

Note: When the OPAnEN is enabled, the GSOPnP must be enabled.

- **CMPC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	CMPPOL	CMPO	CMPEN	GSCMPN0	GSCMPP2	GSCMPP1	GSCMPP0
R/W	—	R/W	R	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 **CMPPOL**: Comparator output polarity Control
    0: Output is not inverted
    1: Output is inverted
    This is the Comparator polarity control bit. If the bit is zero then the comparator output bit, CMPO, will reflect the non-inverted output condition of the comparator. If the bit is high the comparator output bit will be inverted.

Bit 5 **CMPO**: Comparator output bit
    If CMPPOL=0,
     0: CMPINP < CMPINN
     1: CMPINP > CMPINN
    If CMPPOL=1,
     0: CMPINP > CMPINN
     1: CMPINP < CMPINN
    This bit stores the Comparator output bit. The polarity of the bit is determined by the voltages on the comparator inputs and by the condition of the CMPPOL bit.

Bit 4 **CMPEN**: Comparator enable/disable control
    0: Disable
    1: Enable
    When the comparator output is from low to high, the comparator will generate an interrupt.

Bit 3 **GSCMPN0**: CMP selection switch for DAC0O
    0: Off
    1: On

Bit 2 **GSCMPP2**: CMP Selection switch for OP3O
    0: Off
    1: On

Bit 1 **GSCMPP1**: CMP Selection switch for OP2O
    0: Off
    1: On

Bit 0 **GSCMPP0**: CMP Selection switch for OP1O
   0: Off
    1: On

Note: When the CMPEN is enabled, the GSCMPN0 must be enabled. Only one of the GSCMPP2, GSCMPP1 and GSCMPP0 bits can be on each time.

Measurement Timing Control

Timing control is used to periodically start each group of the DAC, DACVREF, OPA and CMP by hardware for application.

The TPRD3~TPRD0 bits are used to set the timing control clock period, with a minimum duration time of 0.5s. The TOPST, TDAVRST, TDAST and TCPST registers can be used to set the number of tick clocks before starting the OPA, DACVREF, DAC and CMP.

The TOPLEN, TDAVRLEN, TDALEN and TCPLEN registers are used to set the turned on duration time of DAC, DACVREF, OPA and CMP. There are four DAC converters, DAC0~DAC3 and three OPA functions, OPA1~OPA3. The TCHS0 and TCHS1 registers can be used to execute timing control for the desired targets. However, if the TCHS0 register is set to start DAC0 and DAC2 for instance, because their start delay time and turned on duration are controlled by TDAST and TDALEN registers respectively, the start delay time and turned on duration of DAC0 and DAC2 are the same, as that of OPAs. When the full period ends, that is, the TPRD count ends, all DAC, OPA , CMP and DACVREF will be forcibly turned off and a new period will be executed again.
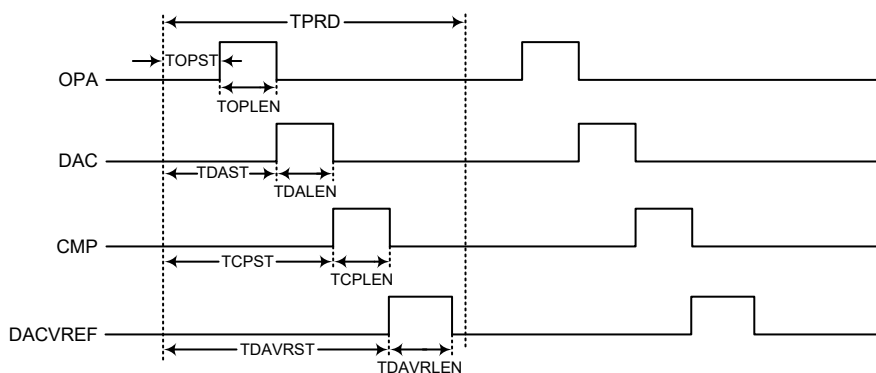

Analog to Digital Converter – ADC

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

A/D Converter Overview

This device contains a 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 channels can be arranged as two single-ended A/D converter input channels or one differential input channel. 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. With high accuracy and performance, the device is very suitable for differential output sensor applications such as weight measurement scales and other related products.
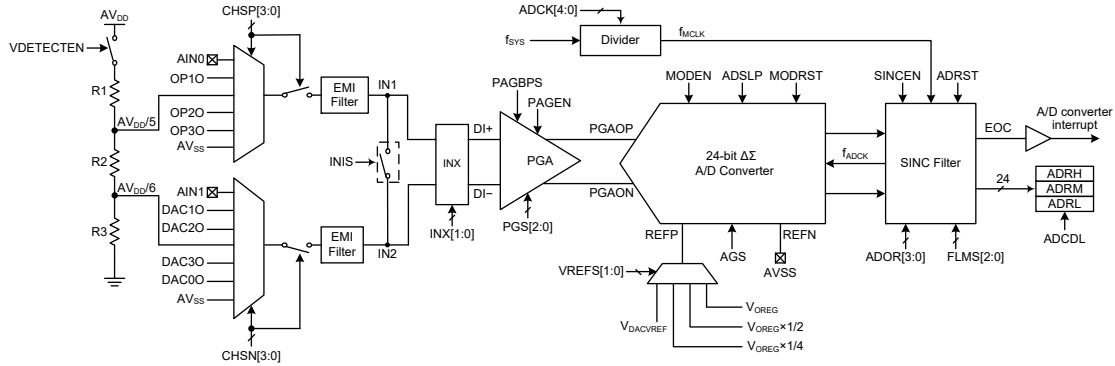
Note: The PGA and A/D converter are supplied by the V_{OREG} and AV_{SS}. The SINC Filter is supplied by the V_{DD} and V_{SS}.

A/D Converter Structure

Internal Power Supply

This device contains an LDO for the regulated power supply. The accompanying block diagram illustrates the basic function operation. The internal LDO can provide a fixed voltage for PGA, A/D converter or external components. There are four LDO voltage levels, 2.4V, 2.6V, 2.2V or 2.3V, 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.
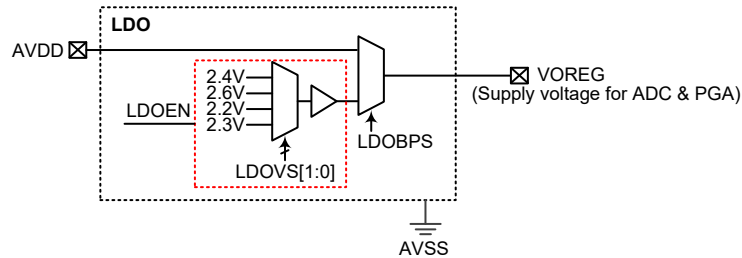
Internal Power Supply Block Diagram

• **PWRC Register**

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

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

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

Bit 6   **LDOSTS**: LDO output pull-low resistor or floating control when LDO is disabled
        0: Pull-low resistor
        1: Floating
        Note: When the LDO is in the bypass mode, this bit can be set high to reduce power consumption.

Bit 5~4   Unimplemented, read as "0"

Bit 3   **VDETECTEN**: Voltage divider $AV_{DD}/5$ and $AV_{DD}/6$ control
      0: Disable
      1: Enable

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

Bit 1~0   **LDOVS1~LDOVS0**: LDO output voltage selection
      00: 2.4V
      01: 2.6V
      10: 2.2V
      11: 2.3V

A/D Converter Register Description

Overall operation of the A/D converter is controlled using several registers. Three read only registers exist to store the A/D converter data 24-bit value. The remaining registers are control registers which setup the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PWRC	LDOEN	LDOSTS	—	—	VDETECTEN	LDOBPS	LDOVS1	LDOVS0
PGAC0	—	VREFS1	VREFS0	—	AGS	PGS2	PGS1	PGS0
PGAC1	—	INIS	INX1	INX0	—	—	PGABPS	PGAEN
PGACS	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
ADRL	D7	D6	D5	D4	D3	D2	D1	D0
ADRM	D15	D14	D13	D12	D11	D10	D9	D8
ADRH	D23	D22	D21	D20	D19	D18	D17	D16
ADCR0	ADRST	—	SINCEN	ADOR3	ADOR2	ADOR1	ADOR0	—
ADCR1	FLMS2	FLMS1	FLMS0	—	—	ADCDL	EOC	—
ADCR2	D7	D6	—	MODEN	D3	D2	D1	D0
ADCR3	MODRST	D6	D5	D4	—	—	—	D0
ADCS	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
SINC3	D7	D6	D5	D4	D3	D2	R_FILSEL	R_CKCHOP

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 resister is used to select the PGA gain, A/D Converter gain and the A/D Converter reference voltage. The PGAC1 register is used to define the input connection, PGA enable/disable control and PGA bypass control. 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 signals, OPA outputs, DAC outputs or internal power supply are actually connected to the internal differential A/D converter.

• **PGAC0 Register**

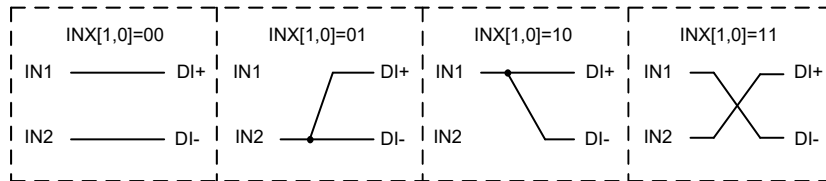| Bit	7	6	5	4	3	2	1	0
Name	—	VREFS1	VREFS0	—	AGS	PGS2	PGS1	PGS0
R/W	—	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	—	0	0	—	0	0	0	0

Bit 7   Unimplemented, read as "0"

Bit 6~5   **VREFS1~VREFS0**: A/D converter reference voltage selection
         00: $V_{DACVREF}$
         01: V_{OREG}
         10: $V_{OREG} \times 1/2$
         11: $V_{OREG} \times 1/4$

Bit 4   Unimplemented, read as "0"

Bit 3   **AGS**: A/D converter PGAOP/PAGON differential input signal gain selection
         0: ADGN=1
         1: ADGN=2

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	—	—	PGABPS	PGAEN
R/W	—	R/W	R/W	R/W	—	—	R/W	R/W
POR	—	0	0	0	—	—	0	0

Bit 7   Unimplemented, read as "0"

Bit 6   **INIS**: Selected inputs, IN1/IN2, internal connection control
         0: Not connected
         1: Connected

Bit 5~4   **INX1~INX0**: Selected inputs, IN1/IN2, and the PGA differential input ends, DI+/DI- connection control bit

Bit 3~2   Unimplemented, read as "0"

Bit 1   **PGABPS**: PGA bypass control
         0: Normal mode
         1: Bypass PGA mode

Bit 0   **PGAEN**: PGA enable/disable control
         0: Disable
         1: Enable
         The PGAEN bit must be enabled before using the PGA function.

• **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
        0000~0111: Reserved
        1000: DAC1O
        1001: DAC2O
        1010: DAC3O
        1011: DAC0O
        1100: $AV_{DD}/6$
        1101: AV_{SS}
        1110: AIN1
        1111: Reserved

Bit 3~0   **CHSP3~CHSP0**: Positive input end IN1 selection
        0000~0111: Reserved
        1000: OP1O
        1001: OP2O
        1010: OP3O
        1011: AV_{SS}
        1100: $AV_{DD}/5$
        1101: AIN0
        1110~1111: Reserved

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 is complete, these registers can be directly read by the microcontroller to obtain the digitised conversion value, D23~D0

• **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, ADCR2, ADCR3, ADCS

To control the function and operation of the A/D converter, several control registers known as ADCR0, ADCR1, ADCR2, ADCR3 and ADCS are provided. These 8-bit registers define functions such as the selection of which oversampling rate is used by the internal A/D converter, the A/D converter clock source as well as controlling the power-up function and modulator control.

- **ADCR0 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADRST	—	SINCEN	ADOR3	ADOR2	ADOR1	ADOR0	—
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	—
POR	0	—	0	0	0	0	0	—

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

Bit 6   Unimplemented, read as "0"

Bit 5   **SINCEN**: SINC function control
         0: Disable
         1: Enable

Bit 4~1   **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
         Others: Reserved

Bit 0   Unimplemented, read as "0"

- **ADCR1 Register**

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

Bit 7~5   **FLMS2~FLMS0**: A/D converter clock f_{ADCK} and divide factor (N) selection
         000: $f_{ADCK}= f_{MCLK}/30$, N=30
         010: $f_{ADCK}= f_{MCLK}/12$, N=12
         Others: Reserved

Bit 4~3   Unimplemented, read as "0"

Bit 2   **ADCDL**: A/D converted data latch function control

       0: Disable data latch function

       1: Enable data latch function

If the A/D converted data latch function is enabled, the latest converted data value will be latched and will not be updated by any subsequent conversion 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 bit 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 1   **EOC**: End of A/D conversion flag

       0: A/D conversion in progress

       1: A/D conversion ended

This bit must be cleared by software.

Bit 0   Unimplemented, read as "0"

• **ADCR2 Register**

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

Bit 7~6   **D7~D6**: Reserved bits, cannot be changed

Bit 5   Unimplemented, read as "0"

Bit 4   **MODEN**: Modulator control

       0: Disable

       1: Enable

Bit 3~0   **D3~D0**: Reserved bits, cannot be changed

• **ADCR3 Register**

Bit	7	6	5	4	3	2	1	0
Name	MODRST	D6	D5	D4	—	—	—	D0
R/W	R/W	R/W	R/W	R/W	—	—	—	R/W
POR	0	1	1	1	—	—	—	1

Bit 7   **MODRST**: Modulator reset control

       0: Normal

       1: Reset

Bit 6~4   **D6~D4**: Reserved bits, cannot be changed

Bit 3~2   Unimplemented, read as "0"

Bit 1   **D0**: Reserved bits, cannot be changed

• **ADCS Register**

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

Bit 7~5   Unimplemented, read as "0"

Bit 4~0   **ADCK4~ADCK0**: A/D converter clock source f_{MCLK} setup

       00000~11110: $f_{MCLK}=f_{SYS}/2/(ADCK[4:0]+1)$

       11111: $f_{MCLK}=f_{SYS}$

SINC Filter Register – SINC3

There is a register related to the SINC Filter control, SINC3. This register is used for the output digital filter selection and the chopper function control.

- **SINC3 Register**

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

Bit 7~2   **D7~D2**: Reserved bits, cannot be changed

Bit 1   **R_FILSEL**: Output digital filter selection
        0: SINC2 filter (R_CKCHOP should be logic low), Data latency = 3 output data clocks
        1: SINC3 filter (R_CKCHOP should be logic high), Data latency = 4 output data clocks

Bit 0   **R_CKCHOP**: System chopper function selection
        0: Enable (R_FILSEL should be logic low)
        1: Disable (R_FILSEL should be logic high)

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

$$= f_{ADCK}/(2 \times OSR)$$

$$= (f_{MCLK}/N)/(2 \times OSR)$$

$$= f_{MCLK} / (N \times 2 \times OSR)$$

Data Rate for SINC3

$$= f_{ADCK}/OSR$$

$$= (f_{MCLK}/N)/OSR$$

$$= f_{MCLK}/(N \times OSR)$$

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

f_{MCLK}: A/D converter clock source, derived from f_{SYS} or $f_{SYS}/2/(ADCK[4:0]+1)$ determined by the ADCK[4:0] bits.

N: A constant divide factor equal to 12 or 30 determined by the FLMS[2:0] bits.

OSR: Oversampling rate determined by the ADOR[3:0] bits.

For example, if a data rate of 8Hz is desired, an f_{MCLK} clock source with a frequency of 4MHz can be selected. Then set the FLMS[2:0] bits to "000" to obtain an "N" equal to 30.

For the SINC2 filter, set the ADOR[3:0] bits 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[3:0] bits to "0001" to select an oversampling rate equal to 16384. Therefore, the Data Rate = 4MHz/(30×16384) = 8Hz.

A/D Converter Operation

To enable the A/D Converter, the first step is to disable the A/D converter power down to make sure the A/D Converter is powered up. The ADRST bit in the ADCR0 register is used to start and reset the A/D converter after power-on. When the microcontroller changes the ADRST bit from low to high and then low again, an analog to digital conversion cycle 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 A/D converter and global 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 "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 newly 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 an internal reference source.

Summary of A/D Conversion Steps

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

- Step 1

  Enable the LDO for PGA and A/D converter power supply.

- Step 2

  Select the PGA, A/D converter gains and reference voltage by PGAC0 register.

- Step 3

  Select the PGA settings for enable/disable control, input connection, input offset and bypass control by PGAC1 register.

- Step 4

  Select the required A/D conversion clock source by correctly programming bits ADCK4~ADCK0 in the ADCS register.

- Step 5

  Select output data rate by configuring the ADOR3~ADOR0 bits in the ADCR0 register and FLMS2~FLMS0 bits in the ADCR1 register.

- Step 6

  Select which channel is to be connected to the internal PGA by correctly programming the CHSP3~CHSP0 and CHSN3~CHSN0 bits which are contained in the PGACS register.

- Step 7

  Reset the A/D by setting the ADRST to high in the ADCR0 register and clearing this bit to zero to release reset status.

- Step 8

  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 9

  To check when the analog to digital conversion process is completed, the EOC bit in the ADCR1 register can be polled. The conversion process is completed 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 A/D converter interrupt is 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 internal circuitry can be switched off to reduce power consumption. 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 value of the differential reference input voltage, $\Delta VR\pm$, selected by the VREFS[1:0] bits in PGAC0 register, this gives a single bit analog input value of $\Delta VR\pm$ divided by 8388608.

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

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], AGS control bits.

  2. ΔSI_I: Differential Input Signal after amplification and offset adjustment.

  3. PGAGN: Programmable Gain Amplifier gain

  4. ADGN: A/D Converter gain

  5. $\Delta DI\pm$: Differential input signal derived from external channels or internal signals

 6. $\Delta VR\pm$: 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
0x7FFFFF	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.
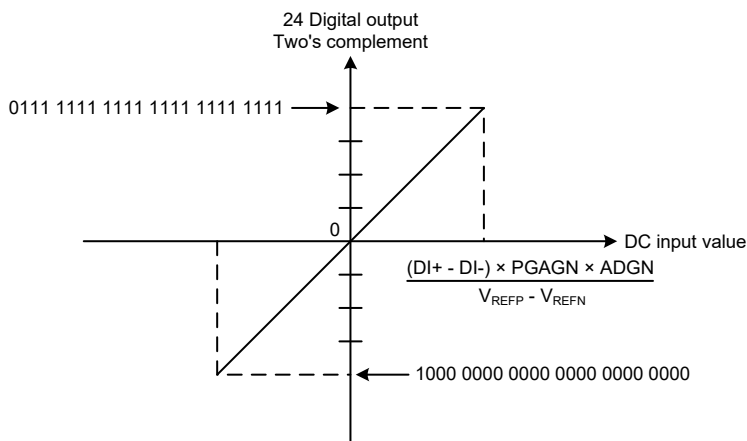
A/D Converted Data

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 converted data 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

Serial Interface Module – SIM

The device contains a Serial Interface Module, which includes both the four-line SPI interface or two-line I²C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I²C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins and therefore the SIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As both interface types share the same pins and registers, the choice of whether the SPI or I²C type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O pins are selected using pull-high control registers when the SIM function is enabled and the corresponding pins are used as SIM input pins.

SPI Interface

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

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

SPI Interface Operation

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


SPI Master/Slave Connection
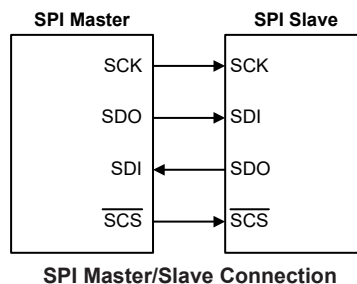
The SPI function in this device offers the following features:

• Full duplex synchronous data transfer

• Both Master and Slave modes

• LSB first or MSB first data transmission modes

- Transmission complete flag
- Rising or falling active clock edge

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

SPI Block Diagram
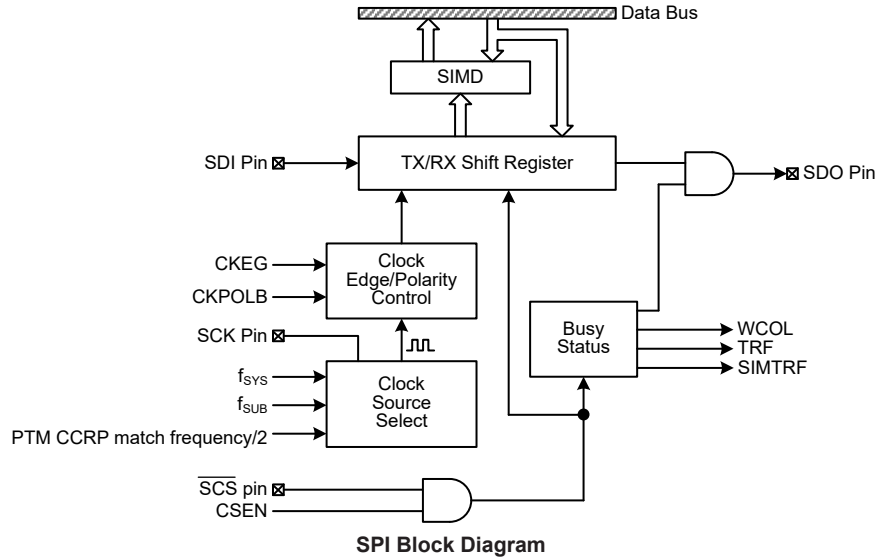
SPI Registers

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

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

SPI Register List

SPI Data Register

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

- **SIMD Register**

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

"x": unknown

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

SPI Control Registers

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

• **SIMC0 Register**

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

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

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

Bit 4   Unimplemented, read as "0"

Bit 3~2   **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
The SIMDEB1~SIMDEB0 bits are only used in the I²C mode and the detailed definition is described in the I²C section.

Bit 1   **SIMEN**: SIM Enable Control
    0: Disable
    1: Enable

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

Bit 0   **SIMICF**: SIM SPI slave mode Incomplete Transfer Flag
    0: SIM SPI slave mode incomplete condition not occurred
    1: SIM SPI slave mode incomplete condition occurred

This bit is only available when the SIM is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SIMEN and CSEN bits both being set to 1 but the $\overline{SCS}$ line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set to 1 together with the TRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the TRF bit will not be set to 1 if the SIMICF bit is set to 1 by software application program.
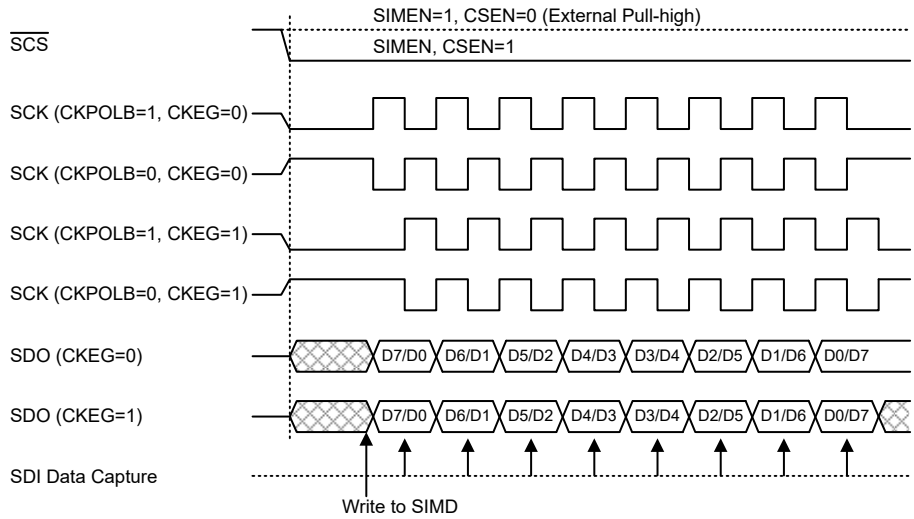
• **SIMC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6   **D7~D6**: Undefined bits

These bits can be read or written by the application program.

Bit 5   **CKPOLB**: SPI clock line base condition selection

0: The SCK line will be high when the clock is inactive

1: The SCK line will be low when the clock is inactive

The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4   **CKEG**: SPI SCK clock active edge type selection

CKPOLB=0

0: SCK is high base level and data capture at SCK rising edge

1: SCK is high base level and data capture at SCK falling edge

CKPOLB=1

0: SCK is low base level and data capture at SCK falling edge

1: SCK is low base level and data capture at SCK rising edge

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

Bit 3   **MLS**: SPI data shift order

0: LSB first

1: MSB first

This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2   **CSEN**: SPI $\overline{\text{SCS}}$ pin control

0: Disable

1: Enable

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

Bit 1   **WCOL**: SPI write collision flag

0: No collision

1: Collision

The WCOL flag is used to detect whether a data collision has occurred or not. If this bit is high, it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. This bit can be cleared by the application program.

Bit 0   **TRF**: SPI Transmit/Receive complete flag

0: SPI data is being transferred

1: SPI data transfer is completed

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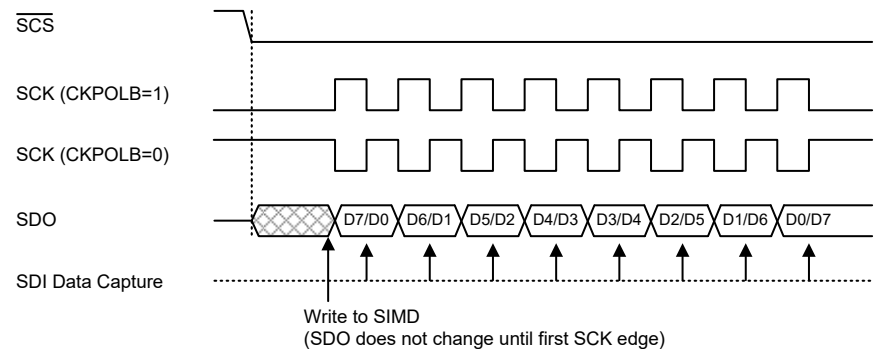
SPI Communication

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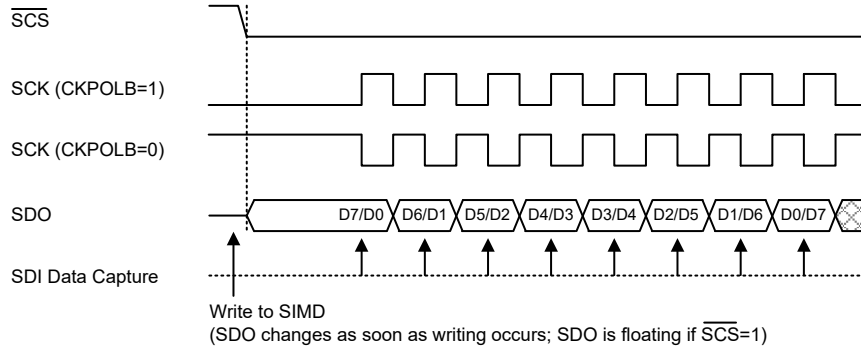
The SPI Master mode will continue to function if the SPI clock is running.

SPI Master Mode Timing

SPI Slave Mode Timing – CKEG=0

SCS

SCK (CKPOLB=1)

SCK (CKPOLB=0)

SDO

SDI Data Capture

Write to SIMD
(SDO changes as soon as writing occurs; SDO is floating if SCS=1)
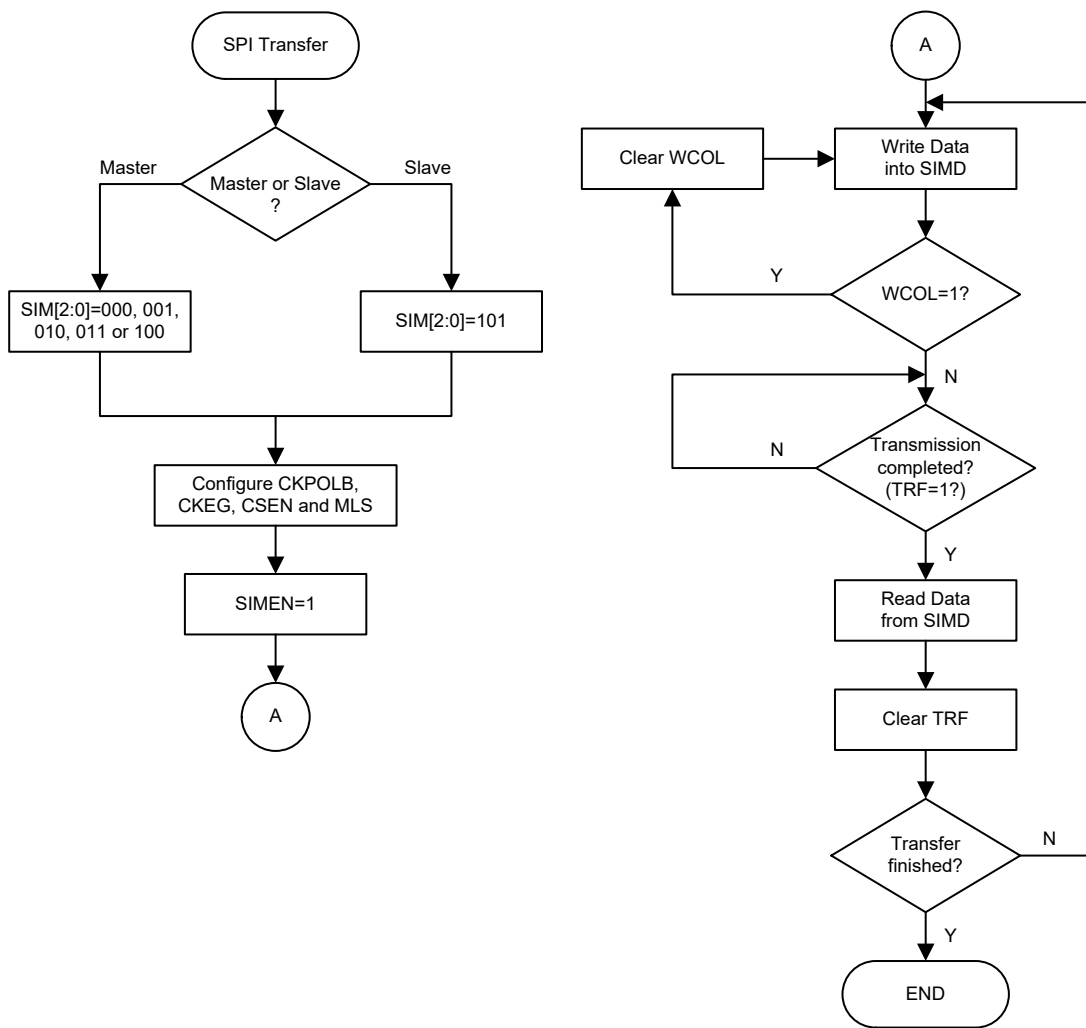
Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always
enabled and ignores the SCS level.

SPI Slave Mode Timing – CKEG=1

SPI Transfer Control Flow Chart

SPI Bus Enable/Disable

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

When the SPI bus is disabled, the SCK, SDI, SDO and $\overline{SCS}$ can become I/O pins or other pin-shared functions using the corresponding pin-shared control bits.

SPI Operation Steps

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

The CSEN bit in the SIMC2 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{SCS}$ line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the $\overline{SCS}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the SDI line in a floating condition and the SDO line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and the $\overline{SCS}$, SDI, SDO and SCK will all become I/O pins or the other functions using the corresponding pin-shared control bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SIMD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode

- Step 1
  Select the SPI Master mode and clock source using the SIM2~SIM0 bits in the SIMC0 control register.

- Step 2
  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave devices.

- Step 3
  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.

- Step 4
  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then use the SCK and SDO lines to output the data. After this, go to step 5.
  For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.

- Step 5
  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.

- Step 6
  Check the TRF bit or wait for an SIM SPI serial bus interrupt.

- Step 7
  Read data from the SIMD register.

- Step 8

  Clear TRF.

- Step 9

  Go to step 4.

Slave Mode

- Step 1

  Select the SPI Slave mode using the SIM2~SIM0 bits in the SIMC0 control register

- Step 2

  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.

- Step 3

  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.

- Step 4

  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and $\overline{SCS}$ signal. After this, go to step 5.

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

- Step 5

  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.

- Step 6

  Check the TRF bit or wait for an SIM SPI serial bus interrupt.

- Step 7

  Read data from the SIMD register.

- Step 8

  Clear TRF.

- Step 9

  Go to step 4.

Error Detection

The WCOL bit in the SIMC2 register is provided to indicate errors during data transfer. The bit is set by the SPI serial Interface but must be cleared by the application program. This bit indicates that a data collision has occurred which happens if a write to the SIMD register takes place during a data transfer operation and will prevent the write operation from continuing.
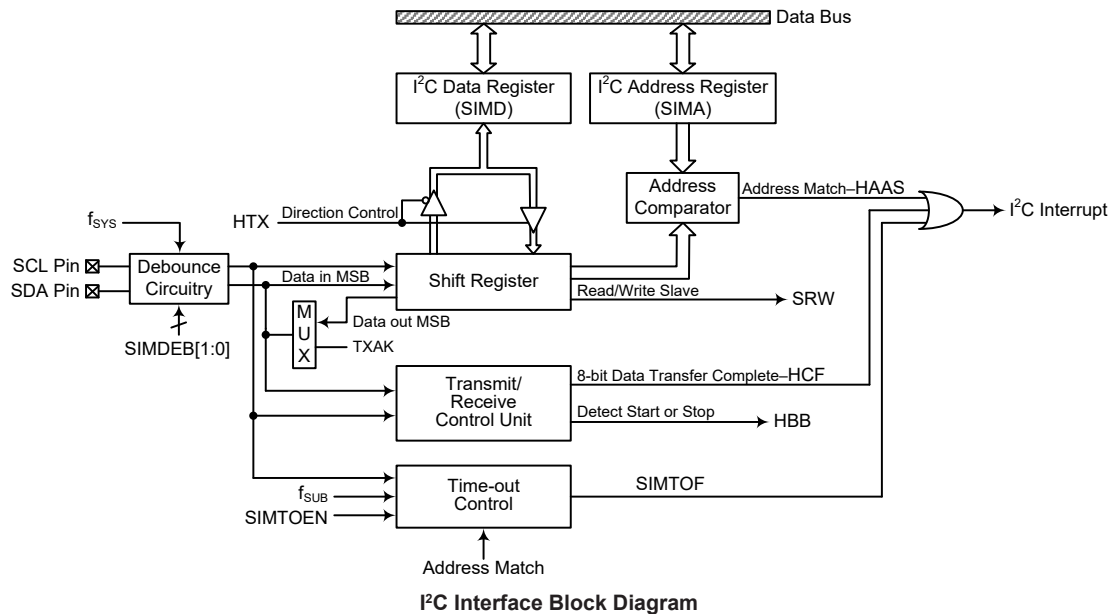
I²C Interface

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

I²C Master/Slave Bus Connection
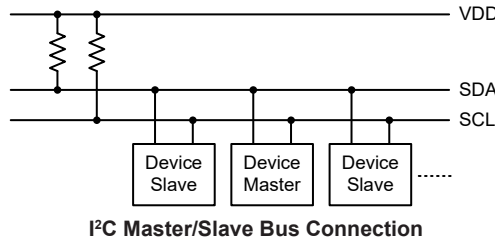
I²C Interface Operation

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

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

I²C Interface Block Diagram

```
┌─────────────────────┐
│    START signal     │
│    from Master      │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│  Send slave address │
│ and R/W bit from    │
│       Master        │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│    Acknowledge      │
│    from slave       │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│   Send data byte    │
│    from Master      │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│    Acknowledge      │
│    from slave       │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│    STOP signal      │
│    from Master      │
└─────────────────────┘
```
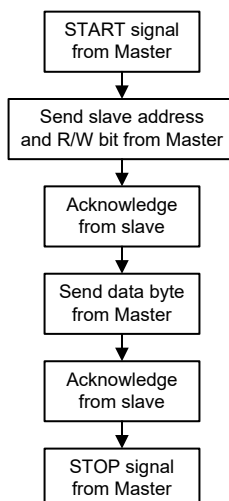
I²C Interface Operation

The SIMDEB1 and SIMDEB0 bits determine the debounce time of the I²C interface. This uses the system clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required I²C data transfer speed, there exists a relationship between the system clock, f_{SYS}, and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

I²C Debounce Time Selection	I²C Standard Mode (100kHz)	I²C Fast Mode (400kHz)
No Debounce	$f_{SYS}>2MHz$	$f_{SYS}>4MHz$
2 system clock debounce	$f_{SYS}>4MHz$	$f_{SYS}>8MHz$
4 system clock debounce	$f_{SYS}>4MHz$	$f_{SYS}>8MHz$

I²C Minimum f_{SYS} Frequency Requirement

I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMTOC, one address register SIMA and one data register, SIMD.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMA	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C Register List

I²C Data Register

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

- **SIMD Register**

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

"x": nknown

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

I²C Address Register

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

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

- **SIMA Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~1   **SIMA6~SIMA0**: I²C slave address
                 SIMA6~SIMA0 is the I²C slave address bit 6~bit 0.
Bit 0   **D0**: Reserved bit, can be read or written

I²C Control Registers

There are three control registers for the I²C interface, SIMC0, SIMC1 and SIMTOC. The register SIMC0 is used to control the enable/disable function and to set select the I²C slave mode and debounce time. The SIMC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, SIMTOC, is used to control the I²C time-out function and is described in the corresponding section.

- **SIMC0 Register**

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

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

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

Bit 4   Unimplemented, read as "0"

Bit 3~2   **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
　　　　　　　　00: No debounce
　　　　　　　　01: 2 system clock debounce
　　　　　　　　1x: 4 system clock debounce
　　　　　　　These bits are used to select the I²C debounce time when the SIM is configured as the I²C interface function by setting the SIM2~SIM0 bits to "110".

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

Bit 0   **SIMICF**: SIM SPI Incomplete Flag
　　　　　　　The SIMICF bit is only used in the SPI mode and the detailed definition is described in the SPI section.

• **SIMC1 Register**

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

Bit 7   **HCF**: I²C Bus data transfer completion flag
　　　　　　　　0: Data is being transferred
　　　　　　　　1: Completion of an 8-bit data transfer
　　　　　　　The HCF flag is the data transfer completion flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated. 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 SIMD 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 data transfer completion flag
　　　　　　　　0: Not address match
　　　　　　　　1: Address match
　　　　　　　The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB**: I²C Bus busy flag
  0: I²C Bus is not busy
  1: I²C Bus is busy

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

Bit 4 **HTX**: I²C slave device transmitter/receiver selection
  0: Slave device is the receiver
  1: Slave device is the transmitter

Bit 3 **TXAK**: I²C bus transmit acknowledge flag
  0: Slave sends acknowledge flag
  1: Slave does not send acknowledge flag

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

Bit 2 **SRW**: I²C slave read/write flag
  0: Slave device should be in receive mode
  1: Slave device should be in transmit mode

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

Bit 1 **IAMWU**: I²C Address Match Wake-Up control
  0: Disable
  1: Enable – must be cleared by the application program after wake-up

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

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

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

I²C Bus Communication

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

- Step 1

    Set the SIM2~SIM0 bits to "110" and SIMEN bit to "1" in the SIMC0 register to enable the I²C bus.

- Step 2

    Write the slave address of the device to the I²C bus address register SIMA.

- Step 3

    Set the SIME interrupt enable bit of the interrupt control register to enable the SIM interrupt.
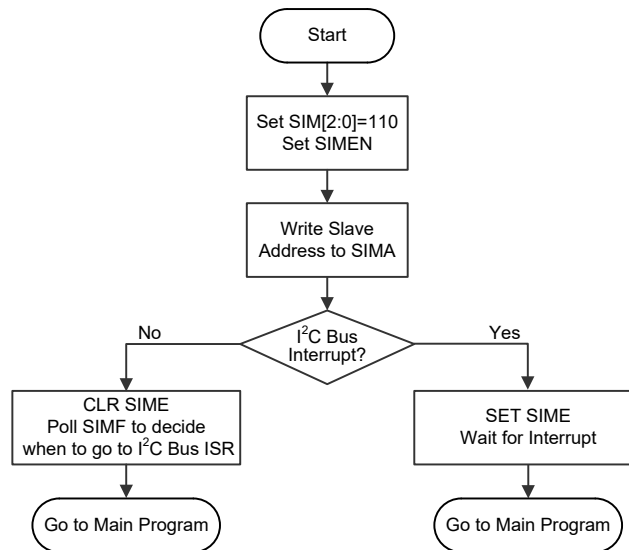
I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

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

I²C Slave Address

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

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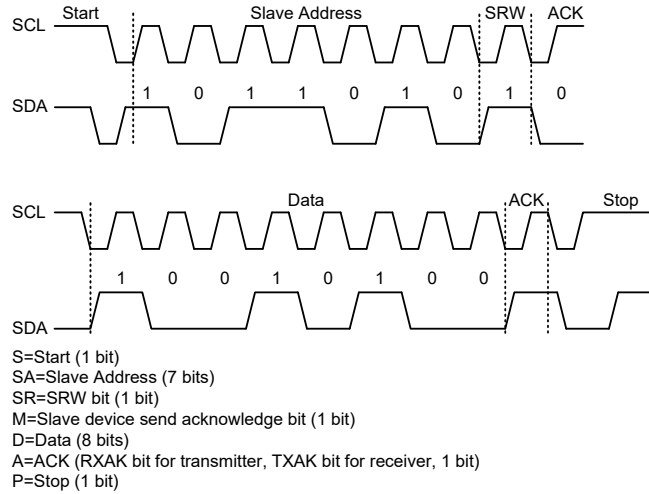
I²C Bus Read/Write Signal

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

I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be cleared to "0".

I²C Bus Data and Acknowledge Signal

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

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

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

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

I²C Communication Timing Diagram
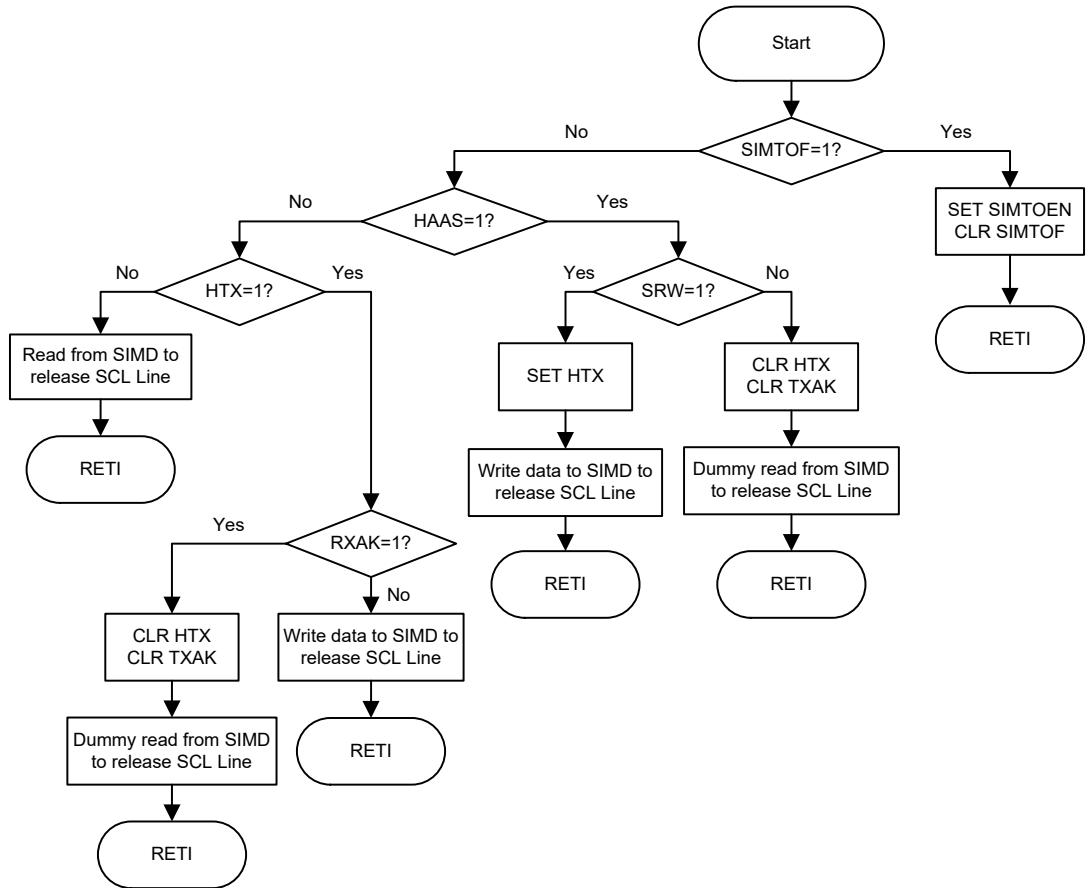
I²C Bus ISR Flow Chart

I²C Time-out Control

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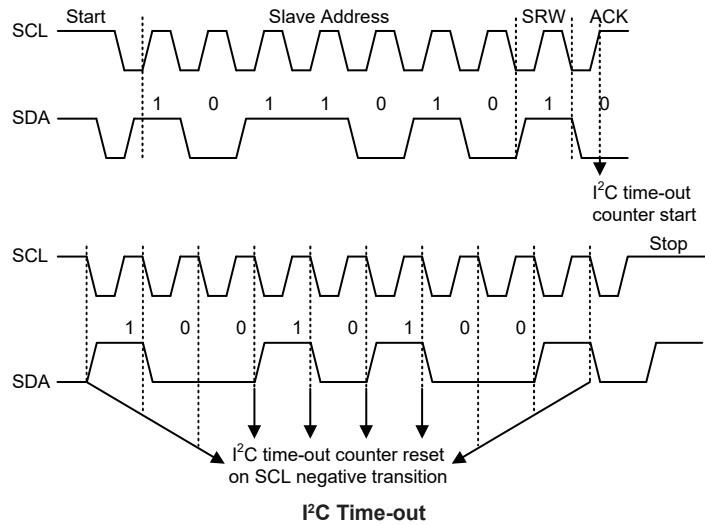
I²C Time-out

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

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

I²C Registers after Time-out

The SIMTOF flag can be cleared by the application program. There are 64 time-out period selections which can be selected using the SIMTOS5~SIMTOS0 bits in the SIMTOC register. The time-out duration is calculated by the formula: $((1{\sim}64) \times (32/f_{SUB}))$. This gives a time-out period which ranges from about 1ms to 64ms.

• **SIMTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7   **SIMTOEN**: SIM I²C Time-out control
           0: Disable
           1: Enable

Bit 6 **SIMTOF**: SIM I²C Time-out flag
    0: No time-out occurred
    1: Time-out occurred

Bit 5~0 **SIMTOS5~SIMTOS0**: SIM I²C Time-out period selection
I²C Time-out clock source is $f_{SUB}/32$.
I²C Time-out period is equal to $(SIMTOS[5:0]+1) \times (32/f_{SUB})$.

16-bit Multiplication Division Unit – MDU

The device has a 16-bit Multiplication Division Unit, MDU, which integrates a 16-bit unsigned multiplier and a 32-bit/16-bit divider. The MDU, in replacing the software multiplication and division operations, can therefore save large amounts of computing time as well as the Program and Data Memory space. It also reduces the overall microcontroller loading and results in the overall system performance improvements.
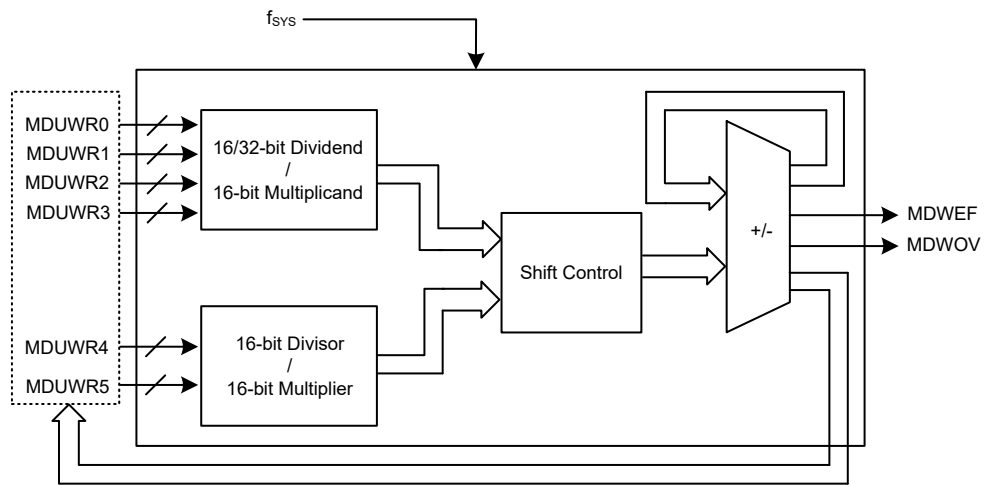
16-bit MDU Block Diagram

MDU Registers

The multiplication and division operations are implemented in a specific way, a specific write access sequence of a series of MDU data registers. The status register, MDUWCTRL, provides the indications for the MDU operation. The data register each is used to store the data regarded as the different operand corresponding to different MDU operations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
MDUWR0	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR1	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR2	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR3	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR4	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR5	D7	D6	D5	D4	D3	D2	D1	D0
MDUWCTRL	MDWEF	MDWOV	—	—	—	—	—	—

MDU Register List

• **MDUWRn Register (n=0~5)**

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

"x": unknown

Bit 7~0　　　**D7~D0**: 16-bit MDU data register n

• **MDUWCTRL Register**

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

Bit 7　　　**MDWEF**: 16-bit MDU error flag
　　　　　　0: Normal
　　　　　　1: Abnormal
　　　　　This bit will be set to 1 if the data register MDUWRn is written or read as the MDU operation is executing. This bit should be cleared to 0 by reading the MDUWCTRL register if it is equal to 1 and the MDU operation is completed.

Bit 6　　　**MDWOV**: 16-bit MDU overflow flag
　　　　　　0: No overflow occurs
　　　　　　1: Multiplication product > FFFFH or Divisor=0
　　　　　When an operation is completed, this bit will be updated by hardware to a new value corresponding to the current operation situation.

Bit 5~0　　　Unimplemented, read as "0"

MDU Operation

For this MDU the multiplication or division operation is carried out in a specific way and is determined by the write access sequence of the six MDU data registers, MDUWR0~MDUWR5. The low byte data, regardless of the dividend, multiplicand, divisor or multiplier, must first be written into the corresponding MDU data register followed by the high byte data. All MDU operations will be executed after the MDUWR5 register is write-accessed together with the correct specific write access sequence of the MDUWRn. Note that it is not necessary to consecutively write data into the MDU data registers but must be in a correct write access sequence. Therefore, a non-write MDUWRn instruction or an interrupt, etc., can be inserted into the correct write access sequence without destroying the write operation. The relationship between the write access sequence and the MDU operation is shown in the following.

• 32-bit/16-bit division operation: Write data sequentially into the six MDU data registers from MDUWR0 to MDUWR5.

• 16-bit/16-bit division operation: Write data sequentially into the specific four MDU data registers in a sequence of MDUWR0, MDUWR1, MDUWR4 and MDUWR5 with no write access to MDUWR2 and MDUWR3.

• 16-bit ×16-bit multiplication operation: Write data sequentially into the specific four MDU data register in a sequence of MDUWR0, MDUWR4, MDUWR1 and MDUWR5 with no write access to MDUWR2 and MDUWR3.

After the specific write access sequence is determined, the MDU will start to perform the corresponding operation. The calculation time necessary for these MDU operations are different. During the calculation time any read/write access to the six MDU data registers is forbidden. After the completion of each operation, it is necessary to check the operation status in the MDUWCTRL register to make sure that whether the operation is correct or not. Then the operation result can be read out from the corresponding MDU data registers in a specific read access sequence if the operation is correctly finished. The necessary calculation time for different MDU operations is listed in the following.

- 32-bit/16-bit division operation: $17 \times t_{SYS}$
- 16-bit/16-bit division operation: $9 \times t_{SYS}$
- 16-bit×16-bit multiplication operation: $11 \times t_{SYS}$

The operation results will be stored in the corresponding MDU data registers and should be read out from the MDU data registers in a specific read access sequence after the operation is completed. Note that it is not necessary to consecutively read data out from the MDU data registers but must be in a correct read access sequence. Therefore, a non-read MDUWRn instruction or an interrupt, etc., can be inserted into the correct read access sequence without destroying the read operation. The relationship between the operation result read access sequence and the MDU operation is shown in the following.

- 32-bit/16-bit division operation: Read the quotient from MDUWR0 to MDUWR3 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit/16-bit division operation: Read the quotient from MDUWR0 and MDUWR1 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit×16-bit multiplication operation: Read the product sequentially from MDUWR0 to MDUWR3.

The overall important points for the MDU read/write access sequence and calculation time are summarized in the following table. Note that the device should not enter the IDLE or SLEEP mode until the MDU operation is totally completed, otherwise the MDU operation will fail.

Operations / Items	32-bit/16-bit Division	16-bit/16-bit Division	16-bit×16-bit Multiplication
Write Sequence First write ↓ ↓ ↓ ↓ Last write	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Dividend Byte 2 written to MDUWR2 Dividend Byte 3 written to MDUWR3 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Multiplicand Byte 0 written to MDUWR0 Multiplier Byte 0 written to MDUWR4 Multiplicand Byte 1 written to MDUWR1 Multiplier Byte 1 written to MDUWR5
Calculation Time	$17 \times t_{SYS}$	$9 \times t_{SYS}$	$11 \times t_{SYS}$
Read Sequence First read ↓ ↓ ↓ ↓ Last read	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Quotient Byte 2 read from MDUWR2 Quotient Byte 3 read from MDUWR3 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Product Byte 0 written to MDUWR0 Product Byte 1 written to MDUWR1 Product Byte 2 written to MDUWR2 Product Byte 3 written to MDUWR3

MDU Operations Summary

Cyclic Redundancy Check – CRC

The Cyclic Redundancy Check, CRC, calculation unit is an error detection technique test algorithm and uses to verify data transmission or storage data correctness. A CRC calculation takes a data stream or a block of data as input and generates a 16-bit output remainder. Ordinarily, a data stream is suffixed by a CRC code and used as a checksum when being sent or stored. Therefore, the received or restored data stream is calculated by the same generator polynomial as described in the following section.

CRC Block Diagram
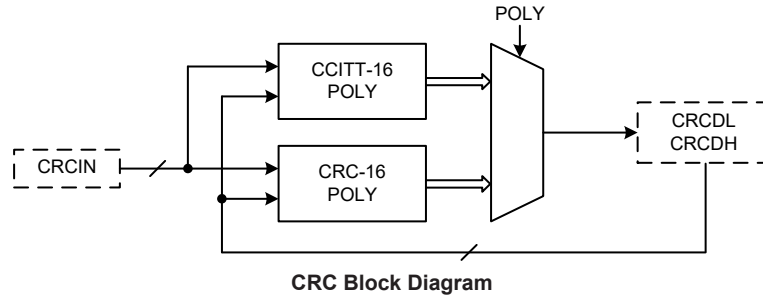
CRC Registers

The CRC generator contains an 8-bit CRC data input register, CRCIN, and a CRC checksum register pair, CRCDH and CRCDL. The CRCIN register is used to input new data and the CRCDH and CRCDL registers are used to hold the previous CRC calculation result. A CRC control register, CRCCR, is used to select which CRC generating polynomial is used.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D15	D14	D13	D12	D11	D10	D9	D8
CRCCR	—	—	—	—	—	—	—	POLY

CRC Register List

• **CRCIN Register**

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

Bit 7~0   **D7~D0**: CRC input data register

• **CRCDL Register**

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

Bit 7~0   **D7~D0**: 16-bit CRC checksum low byte data register

- **CRCDH Register**

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

Bit 7~0   **D15~D8**: 16-bit CRC checksum high byte data register

- **CRCCR Register**

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

Bit 7~1   Unimplemented, read as "0"

Bit 0   **POLY**: 16-bit CRC generating polynomial selection
   0: CRC-CCITT: $X^{16}+X^{12}+X^5+1$
   1: CRC-16: $X^{16}+X^{15}+X^2+1$

CRC Operation

The CRC generator provides the 16-bit CRC result calculation based on the CRC16 and CCITT CRC16 polynomials. In this CRC generator, there are only these two polynomials available for the numeric values calculation. It can not support the 16-bit CRC calculations based on any other polynomials.

The following two expressions can be used for the CRC generating polynomial which is determined using the POLY bit in the CRC control register, CRCCR. The CRC calculation result is called as the CRC checksum, CRCSUM, and stored in the CRC checksum register pair, CRCDH and CRCDL.

- CRC-CCITT: $X^{16}+X^{12}+X^5+1$
- CRC-16: $X^{16}+X^{15}+X^2+1$

CRC Computation

Each write operation to the CRCIN register creates a combination of the previous CRC value stored in the CRCDH and CRCDL registers and the new data input. The CRC unit calculates the CRC data register value is based on byte by byte. It will take one MCU instruction cycle to calculate the CRC checksum.

CRC Calculation Procedures

1. Clear the checksum register pair, CRCDH and CRCDL.

2. Execute an "Exclusive OR" operation with the 8-bit input data byte and the 16-bit CRCSUM high byte. The result is called the temporary CRCSUM.

3. Shift the temporary CRCSUM value left by one bit and move a "0" into the LSB.

4. Check the shifted temporary CRCSUM value after procedure 3.

   If the MSB is 0, then this shifted temporary CRCSUM will be considered as a new temporary CRCSUM.

   Otherwise, execute an "Exclusive OR" operation with the shifted temporary CRCSUM in procedure 3 and a data "8005H". Then the operation result will be regarded as the new temporary CRCSUM.

   Note that the data to be perform an "Exclusive OR" operation is "8005H" for the CRC-16 polynomial while for the CRC-CCITT polynomial the data is "1021H".

5. Repeat the procedure 3 ~ procedure 4 until all bits of the input data byte are completely calculated.

6. Repeat the procedure 2~ procedure 5 until all of the input data bytes are completely calculated. Then, the latest calculated result is the final CRC checksum, CRCSUM.

CRC Calculation Examples

• Write 1 byte input data into the CRCIN register and the corresponding CRC checksum are individually calculated as the following table shown.

CRC Data Input CRC Polynomial	00H	01H	02H	03H	04H	05H	06H	07H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	0000H	1021H	2042H	3063H	4084H	50A5H	60C6H	70E7H
CRC-16 ($X^{16}+X^{15}+X^2+1$)	0000H	8005H	800FH	000AH	801BH	001EH	0014H	8011H

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before each CRC input data is written into the CRCIN register.

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

CRC Data Input CRC Polynomial	CRCIN = 78H→56H→34H→12H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	(CRCDH, CRCDL) = FF9FH→BBC3H→A367H→D0FAH
CRC-16 ($X^{16}+X^{15}+X^2+1$)	(CRCDH, CRCDL) = 0110h→91F1h→F2DEh→5C43h

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before the sequential CRC data input operation.

Program Memory CRC Checksum Calculation Example

1. Clear the checksum register pair, CRCDH and CRCDL.

2. Select the CRC-CCITT or CRC-16 polynomial as the generating polynomial using the POLY bit in the CRCCR register.

3. Execute the table read instruction to read the program memory data value.

4. Write the table data low byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.

5. Write the table data high byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.

6. Repeat the procedure 3 ~ procedure 5 to read the next program memory data value and execute the CRC calculation until all program memory data are read followed by the sequential CRC calculation. Then the value in the CRC checksum register pair is the final CRC calculation result.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions. The external interrupts are generated by the action of the external INT0~INT3 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, EEPROM, SIM, comparator 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 Structure
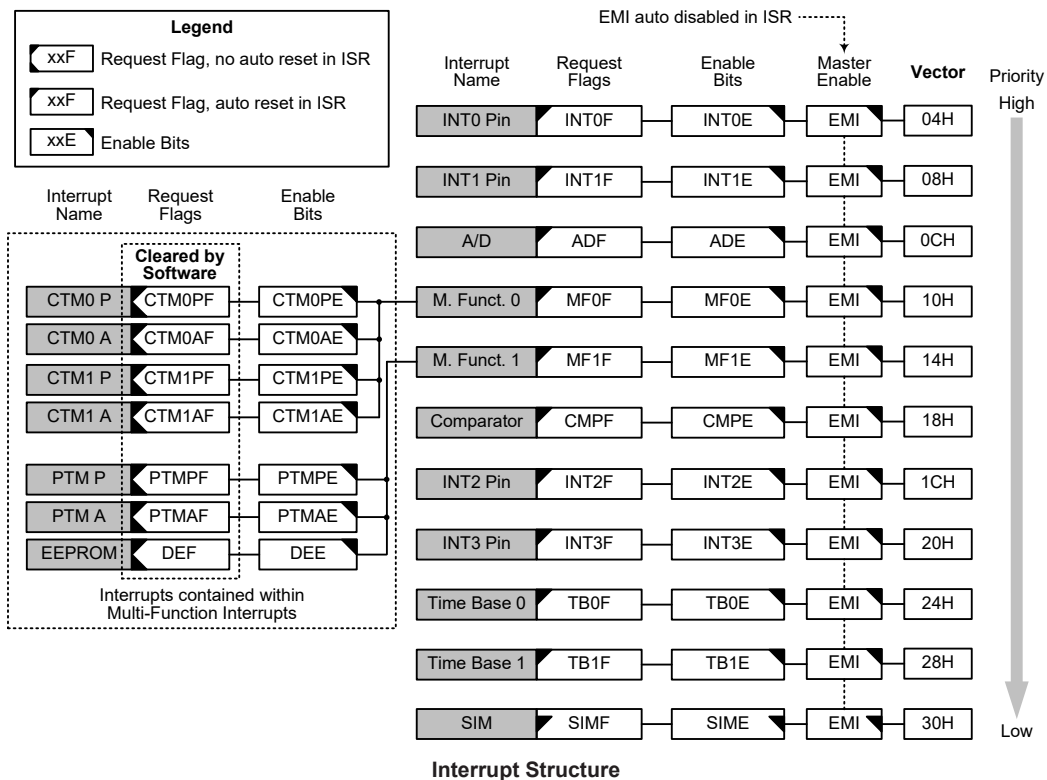
Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers depends upon the device chosen but fall into three categories. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0~MFI1 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

Each register contains a number of enable bits to enable or disable individual interrupts as well as interrupt flags to I ndicate the presence of an interrupt request. The naming convention of these follows a specific pattern. First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an "E" for enable/disable bit or "F" for request flag.

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pins	INTnE	INTnF	n=0~3
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~1
Time Base	TBnE	TBnF	n=0~1
SIM Interface	SIME	SIMF	—
Comparator	CMPE	CMPF	—
EEPROM	DEE	DEF	—
CTM	CTMnPE	CTMnPF	n=0~1
	CTMnAE	CTMnAF	
PTM	PTMPE	PTMPF	—
	PTMAE	PTMAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
INTC1	INT2F	CMPF	MF1F	MF0F	INT2E	CMPE	MF1E	MF0E
INTC2	—	TB1F	TB0F	INT3F	—	TB1E	TB0E	INT3E
INTC3	—	—	—	SIMF	—	—	—	SIME
MFI0	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
MFI1	—	DEF	PTMAF	PTMPF	—	DEE	PTMAE	PTMPE

Interrupt Register List

- INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6   **INT3S1~INT3S0**: Interrupt edge control for INT3 pin
    00: Disable
    01: Rising edge
    10: Falling edge
    11: Rising and falling edges

Bit 5~4   **INT2S1~INT2S0**: Interrupt edge control for INT2 pin
    00: Disable
    01: Rising edge
    10: Falling edge
    11: Rising and falling edges

Bit 3~2   **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
    00: Disable
    01: Rising edge
    10: Falling edge
    11: Rising and falling edges

Bit 1~0   **INT0S1~INT0S0**: Interrupt edge control for INT0 pin
    00: Disable
    01: Rising edge
    10: Falling edge
    11: Rising and falling edges

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7   Unimplemented, read as "0"

Bit 6   **ADF**: A/D Converter interrupt request flag
        0: No request
        1: Interrupt request

Bit 5   **INT1F**: INT1 interrupt request flag
        0: No request
        1: Interrupt request

Bit 4   **INT0F**: INT0 interrupt request flag
        0: No request
        1: Interrupt request

Bit 3   **ADE**: A/D Converter interrupt control
        0: Disable
        1: Enable

Bit 2   **INT1E**: INT1 interrupt control
        0: Disable
        1: Enable

Bit 1   **INT0E**: INT0 interrupt control
        0: Disable
        1: Enable

Bit 0   **EMI**: Global interrupt control
        0: Disable
        1: Enable

• **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	INT2F	CMPF	MF1F	MF0F	INT2E	CMPE	MF1E	MF0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7   **INT2F**: INT2 interrupt request flag
        0: No request
        1: Interrupt request

Bit 6   **CMPF**: Comparator interrupt request flag
        0: No request
        1: Interrupt request

Bit 5   **MF1F**: Multi-function 1 interrupt request flag
        0: No request
        1: Interrupt request

Bit 4   **MF0F**: Multi-function 0 interrupt request flag
        0: No request
        1: Interrupt request

Bit 3   **INT2E**: INT2 interrupt control
        0: Disable
        1: Enable

Bit 2   **CMPE**: Comparator interrupt control
        0: Disable
        1: Enable

Bit 1   **MF1E**: Multi-function 1 interrupt control
                    0: Disable
                    1: Enable

Bit 0   **MF0E**: Multi-function 0 interrupt control
                    0: Disable
                    1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TB1F	TB0F	INT3F	—	TB1E	TB0E	INT3E
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7   Unimplemented, read as "0"

Bit 6   **TB1F**: Time Base 1 interrupt request flag
                    0: No request
                    1: Interrupt request

Bit 5   **TB0F**: Time Base 0 interrupt request flag
                    0: No request
                    1: Interrupt request

Bit 4   **INT3F**: INT3 interrupt request flag
                    0: No request
                    1: Interrupt request

Bit 3   Unimplemented, read as "0"

Bit 2   **TB1E**: Time Base 1 interrupt control
                    0: Disable
                    1: Enable

Bit 1   **TB0E**: Time Base 0 interrupt control
                    0: Disable
                    1: Enable

Bit 0   **INT3E**: INT3 interrupt control
                    0: Disable
                    1: Enable

• **INTC3 Register**

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

Bit 7~5   Unimplemented, read as "0"

Bit 4   **SIMF**: SIM interrupt request flag
                    0: No request
                    1: Interrupt request

Bit 3~1   Unimplemented, read as "0"

Bit 0   **SIME**: SIM interrupt control
                    0: Disable
                    1: Enable

• **MFI0 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
    Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 6   **CTM1PF**: CTM1 Comparator P match interrupt request flag
      0: No request
      1: Interrupt request
    Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 5   **CTM0AF**: CTM0 Comparator A match interrupt request flag
      0: No request
      1: Interrupt request
    Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4   **CTM0PF**: CTM0 Comparator P match interrupt request flag
      0: No request
      1: Interrupt request
    Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3   **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

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	DEF	PTMAF	PTMPF	—	DEE	PTMAE	PTMPE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7   Unimplemented, read as "0"

Bit 6   **DEF**: Data EEPROM interrupt request flag
      0: No request
      1: Interrupt request
    Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 5   **PTMAF**: PTM Comparator A match interrupt request flag
             0: No request
             1: Interrupt request
          Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4   **PTMPF**: PTM Comparator P match interrupt request flag
             0: No request
             1: Interrupt request
          Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3   Unimplemented, read as "0"

Bit 2   **DEE**: Data EEPROM interrupt control
             0: Disable
             1: Enable

Bit 1   **PTMAE**: PTM Comparator A match interrupt control
             0: Disable
             1: Enable

Bit 0   **PTMPE**: PTM Comparator P match interrupt control
             0: Disable
             1: Enable

Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A or A/D conversion completion, etc, the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

Once an interrupt subroutine is serviced, all other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the 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~INT3. An external interrupt request will take place when the external interrupt request flags, INT0F~INT3F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT3E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT3F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

A/D Converter Interrupt

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 Interrupts

Within the device there are two Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts and EEPROM interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags MFnF is set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to the relevant Multi-function interrupt vector will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function Interrupt request flags will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupts will not be automatically reset and must be manually reset by the application program.

Timer Module Interrupts

The Compact and Periodic type TMs each has two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. For all of the TM Types there are two interrupt request flags and two interrupt enable bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector location, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

EEPROM Interrupt

The EEPROM Interrupt is contained within the Multi-function Interrupt. An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM erase or write cycle ends. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, EEPROM Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and an EEPROM erase or write cycle ends, a subroutine call to the Multi-function Interrupt vector will take place. When the EEPROM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the Multi-function interrupt request flag will be automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signals in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically cleared, the EMI bit will also be automatically cleared to disable other interrupts.

The purpose of the Time Base Interrupts is to provide an interrupt signal at fixed time periods. Its clock source, f_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock cource that generate f_{PSC}, which in turn controls the Time Base interrupt period, is selected using the CLKSEL[1:0] bits in the PSCR register.

Time Base Interrupts
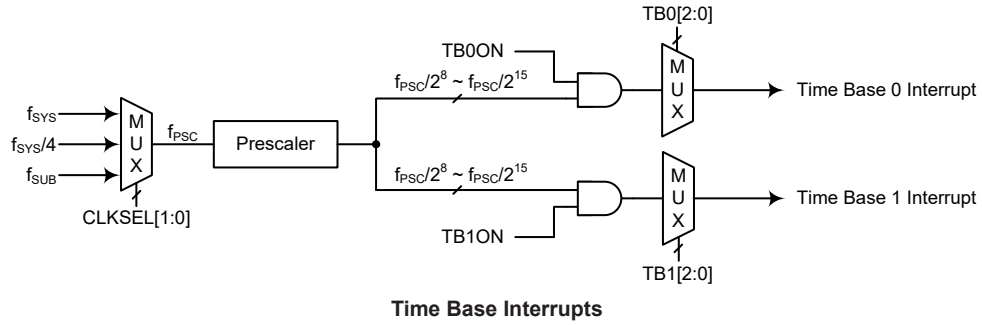
• PSCR Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL1	CLKSEL0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2   Unimplemented, read as "0"

Bit 1~0   **CLKSEL1~CLKSEL0**: Prescaler clock source f_{PSC} selection
      00: f_{SYS}
      01: $f_{SYS}/4$
      10/11: f_{SUB}

• TB0C Register

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7   **TB0ON**: Time Base 0 Enable Control
      0: Disable
      1: Enable

Bit 6~3   Unimplemented, read as "0"

Bit 2~0   **TB02~TB00**: Time Base 0 time-out period selection
      000: $2^8/f_{PSC}$
      001: $2^9/f_{PSC}$
      010: $2^{10}/f_{PSC}$
      011: $2^{11}/f_{PSC}$
      100: $2^{12}/f_{PSC}$
      101: $2^{13}/f_{PSC}$
      110: $2^{14}/f_{PSC}$
      111: $2^{15}/f_{PSC}$

• TB1C Register

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	—	—	—	—	TB12	TB11	TB10
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7   **TB1ON**: Time Base 1 Enable Control
      0: Disable
      1: Enable

Bit 6~3   Unimplemented, read as "0"

Bit 2~0　　**TB12~TB10**: Time Base 1 time-out period selection

　　　　000: $2^8/f_{PSC}$
　　　　001: $2^9/f_{PSC}$
　　　　010: $2^{10}/f_{PSC}$
　　　　011: $2^{11}/f_{PSC}$
　　　　100: $2^{12}/f_{PSC}$
　　　　101: $2^{13}/f_{PSC}$
　　　　110: $2^{14}/f_{PSC}$
　　　　111: $2^{15}/f_{PSC}$

Serial Interface Module Interrupt

The Serial Interface Module Interrupt is also known as the SIM interrupt. A SIM Interrupt request will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface, an I²C slave address match or I²C bus time-out occurs. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and SIM Interrupt enable bit, SIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the corresponding SIM Interrupt vector, will take place. When the interrupt is serviced, the SIMF flag will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

Comparator Interrupt

The comparator interrupt is controlled by the internal comparator in the measurement circuit. A comparator interrupt request will take place when the comparator interrupt request flag, CMPF, is set, which occurs when the specified OPA input signal is detected. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and comparator interrupt enable bit, CMPE, must first be set. When the interrupt is enabled, the stack is not full and the comparator inputs generate a comparator output transition, a subroutine call to the comparator interrupt vector, will take place. When the interrupt is serviced, the comparator interrupt request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program.

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MFnF, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

It is recommended that programs do not use the "CALL" instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a CALL subroutine is executed in the interrupt subroutine.

Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Option	
1	HIRC Frequency Selection – f_{HIRC}: 4MHz, 8MHz or 12MHz

Note: When the HIRC has been configured at a frequency shown in this table, the HIRC1 and HIRC0 bits should also be setup to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Application Circuits
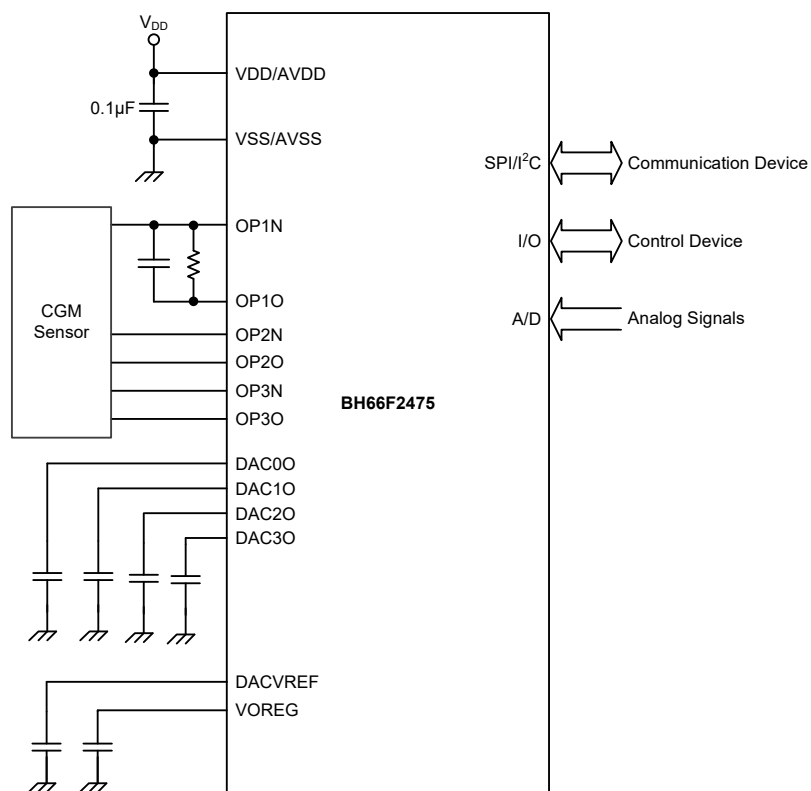
Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of several kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

Table Conventions

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV, SC
ADDM A,[m]	Add ACC to Data Memory	1[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	1[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	1[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	1[Note]	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1[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	1[Note]	Z
ORM A,[m]	Logical OR ACC to Data Memory	1[Note]	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1[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	1[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	1[Note]	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1[Note]	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1[Note]	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1[Note]	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1[Note]	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1[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			
LRRA [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 $ACC \leftarrow ACC \ ''AND'' \ [m]$

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 $ACC \leftarrow ACC \ ''AND'' \ x$

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 $[m] \leftarrow ACC \ ''AND'' \ [m]$

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 ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ x
Affected flag(s)	Z

ORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″OR″ [m]
Affected flag(s)	Z

RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None

RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None

RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None

RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← [m].7
Affected flag(s)	None

RLA [m]	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← [m].7
Affected flag(s)	None

RLC [m]	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← C C ← [m].7
Affected flag(s)	C

RLCA [m]	Rotate Data Memory left through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← C C ← [m].7
Affected flag(s)	C

RR [m]	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← [m].0
Affected flag(s)	None

RRA [m]	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [m].(i+1); (i=0~6) ACC.7 ← [m].0
Affected flag(s)	None

RRC [m]	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← C C ← [m].0
Affected flag(s)	C

RRCA [m]	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [m].(i+1); (i=0~6) ACC.7 ← C C ← [m].0
Affected flag(s)	C

SBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

SBC A, x	Subtract immediate data from ACC with Carry
Description	The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

SBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

SDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] − 1 Skip if [m]=0
Affected flag(s)	None

SDZA [m]	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	ACC ← [m] − 1 Skip if ACC=0
Affected flag(s)	None

SET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	[m] ← FFH
Affected flag(s)	None

SET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	[m].i ← 1
Affected flag(s)	None

SIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] + 1 Skip if [m]=0
Affected flag(s)	None

SIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] + 1 Skip if ACC=0
Affected flag(s)	None

SNZ [m].i	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None

SNZ [m]	Skip if Data Memory is not 0
Description	The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m]≠ 0
Affected flag(s)	None

SUB A,[m]	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m]
Affected flag(s)	OV, Z, AC, C, SC, CZ

SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m]
Affected flag(s)	OV, Z, AC, C, SC, CZ

SUB A,x	Subtract immediate data from ACC
Description	The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − x
Affected flag(s)	OV, Z, AC, C, SC, CZ

SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0 ↔ [m].7~[m].4
Affected flag(s)	None

SWAPA [m]	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3~ACC.0 ← [m].7~[m].4 ACC.7~ACC.4 ← [m].3~[m].0
Affected flag(s)	None

SZ [m]	Skip if Data Memory is 0
Description	The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m]=0
Affected flag(s)	None

SZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] Skip if [m]=0
Affected flag(s)	None

SZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if [m].i=0
Affected flag(s)	None

TABRD [m]	Read table (specific page) to TBLH and Data Memory
Description	The low byte of the program code (specific page) addressed by the table pointer (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

TABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

ITABRD [m]	Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

ITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

XORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″XOR″ [m]
Affected flag(s)	Z

XOR A,x	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ x
Affected flag(s)	Z

Extended Instruction Definition

The extended instructions are used to directly access the data stored in any data memory sections.

LADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC + [m] + C$
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	$ACC \leftarrow ACC\ ''AND''\ [m]$
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	$[m] \leftarrow ACC\ ''AND''\ [m]$
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 ← [m]

Affected flag(s) None

LMOV [m],A   Move ACC to Data Memory

Description   The contents of the Accumulator are copied to the specified Data Memory.

Operation   [m] ← ACC

Affected flag(s) None

LOR A,[m]   Logical OR Data Memory to ACC

Description   Data in the Accumulator and the specified Data Memory perform a bitwise
                 logical OR operation. The result is stored in the Accumulator.

Operation   ACC ← ACC ″OR″ [m]

Affected flag(s) Z

LORM A,[m]   Logical OR ACC to Data Memory

Description   Data in the specified Data Memory and the Accumulator perform a bitwise logical OR
                 operation. The result is stored in the Data Memory.

Operation   [m] ← ACC ″OR″ [m]

Affected flag(s) Z

LRL [m]   Rotate Data Memory left

Description   The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation   [m].(i+1) ← [m].i; (i=0~6)
                 [m].0 ← [m].7

Affected flag(s) None

LRLA [m]   Rotate Data Memory left with result in ACC

Description   The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
                 The rotated result is stored in the Accumulator and the contents of the Data Memory remain
                 unchanged.

Operation   ACC.(i+1) ← [m].i; (i=0~6)
                 ACC.0 ← [m].7

Affected flag(s) None

LRLC [m]   Rotate Data Memory left through Carry

Description   The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7
                 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation   [m].(i+1) ← [m].i; (i=0~6)
                 [m].0 ← C
                 C ← [m].7

Affected flag(s) C

LRLCA [m]   Rotate Data Memory left through Carry with result in ACC

Description   Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the
                 Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the
                 Accumulator and the contents of the Data Memory remain unchanged.

Operation   ACC.(i+1) ← [m].i; (i=0~6)
                 ACC.0 ← C
                 C ← [m].7

Affected flag(s) C

LRR [m]	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← [m].0
Affected flag(s)	None

LRRA [m]	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [m].(i+1); (i=0~6) ACC.7 ← [m].0
Affected flag(s)	None

LRRC [m]	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← C C ← [m].0
Affected flag(s)	C

LRRCA [m]	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [m].(i+1); (i=0~6) ACC.7 ← C C ← [m].0
Affected flag(s)	C

LSBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a 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] $\neq$ 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 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 for FP0.25mm) Outline Dimensions
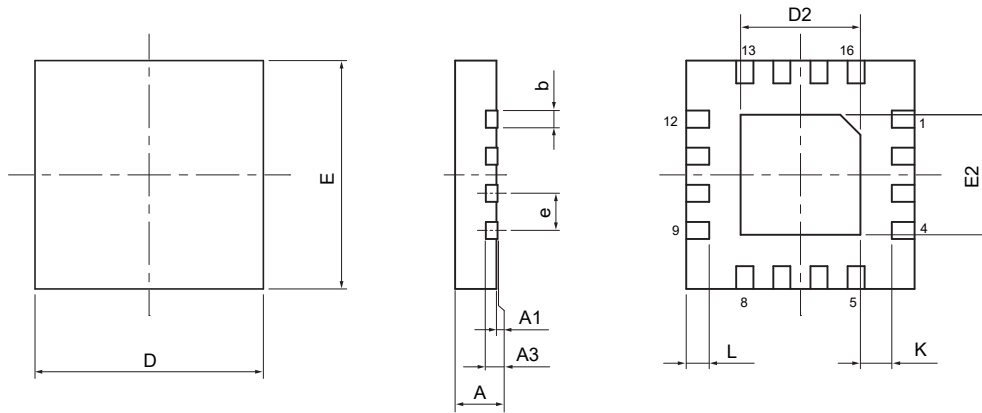
Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	0.008 REF		
b	0.007	0.010	0.012
D	0.118 BSC		
E	0.118 BSC		
e	0.020 BSC		
D2	0.063	—	0.069
E2	0.063	—	0.069
L	0.008	0.010	0.012
K	0.008	—	—

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	—	—

SAW Type 24-pin QFN (3mm×3mm×0.55mm) Outline Dimensions
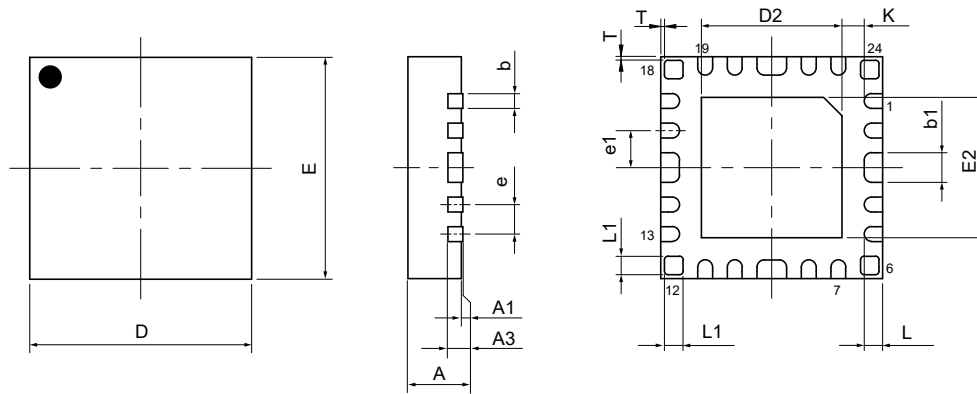
Symbol	Dimensions in inch		
	Min.	**Nom.**	**Max.**
A	0.020	0.022	0.024
A1	0.000	0.001	0.002
A3	0.006 REF		
b	0.006	0.008	0.010
b1	0.014	0.016	0.018
D	0.118 BSC		
E	0.118 BSC		
e	0.016 BSC		
e1	0.020 BSC		
D2	0.073	—	0.077
E2	0.073	—	0.077
L	0.006	0.010	0.014
L1	0.008	0.010	0.012
K	0.008	—	—
T	0.000	0.002	0.004

Symbol	Dimensions in mm		
	Min.	**Nom.**	**Max.**
A	0.50	0.55	0.60
A1	0.00	0.02	0.05
A3	0.15 REF		
b	0.15	0.20	0.25
b1	0.35	0.40	0.45
D	3.00 BSC		
E	3.00 BSC		
e	0.40 BSC		
e1	0.50 BSC		
D2	1.85	—	1.95
E2	1.85	—	1.95
L	0.15	0.25	0.35
L1	0.20	0.25	0.30
K	0.20	—	—
T	0.00	0.05	0.10