Induction Cooker Flash MCU

HT45F0005A

Revision: V1.00 Date: March 15, 2024

www.holtek.com

Table of Contents

Features

CPU Features

- Operating voltage
  - f_{SYS}=16MHz: 3.3V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 16MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 4K×16
- Data Memory: 512×8
- True EEPROM Memory: 128×8
- Watchdog Timer function
- 17 bidirectional I/O lines
- 14 external channel 12-bit resolution A/D converter with Internal Reference Voltage V_{BG}
- 9-bit programmable pulse generator
  - Pulse width limit function
  - Two sets of 9-bit counter preload registers and two sets of 9-bit counter approach registers
  - Non-retrigger control from 8-bit Timer/Event Counter 2
  - Retrigger control from 8-bit Timer/Event Counter 3
  - Active high pulse, active low pulse, forced low or forced high output
- Four 8-bit programmable Timer/Event Counters
  - Timer/Event Counter 0/1 can be configured to count synchronism pulse number or measure synchronism pulse high or low period
  - Timer/Event Counter 2 can be configured to implement PPG non-retrigger function
  - Timer/Event Counter 3 can be configured to implement PPG retrigger function
- 6 comparator functions, five of which are used for Over Voltage Protection functions
- Single Operational Amplifier function – OPAMP
- Peripheral clock output
- I^2C Interface, available for Slave Mode
- Integrated 16-bit Cyclic Redundancy Check function – CRC
- Low voltage reset function
- Low voltage detect function
- Package type: 20-pin SOP

General Description

The device is a Flash Memory type 8-bit high performance RISC architecture microcontroller especially designed for induction cooker applications.

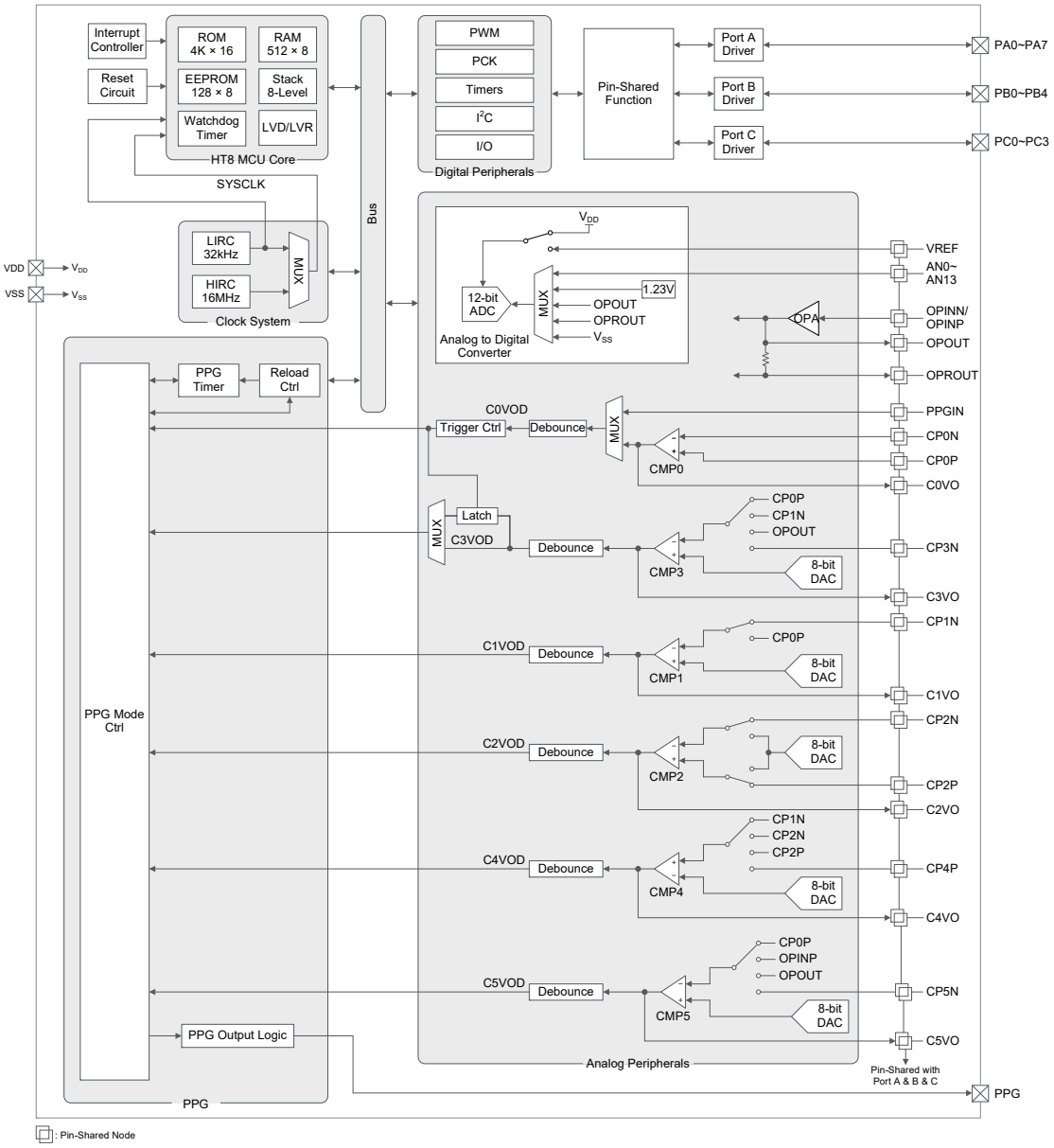
For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 12-bit A/D converter, an operational amplifier and six comparators, five of which are used for Over Voltage Protection function. Easy communication with the outside world is provided using the internal I^2C interface function. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

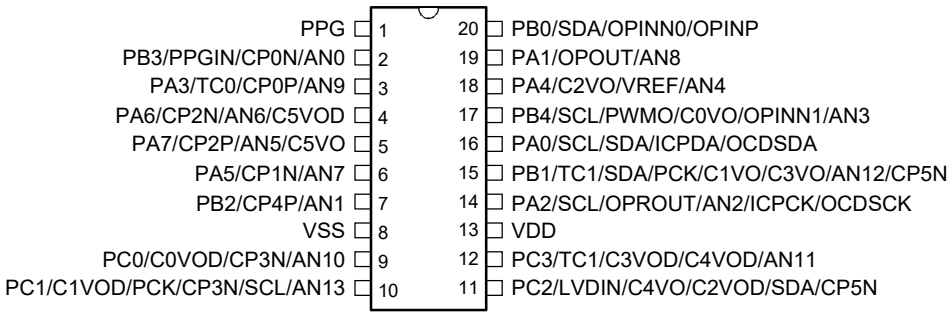
The device also includes fully integrated high and low speed oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation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

The inclusion of flexible I/O programming features, Timers, a Programmable Pulse Generator, a Peripheral Clock Output along with many other features ensure that the device will find excellent use in induction cooker applications.

Block Diagram



Pin Assignment



HT45F0005A/HT45V0005A
20 SOP-A

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

2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the HT45V0005A device which is the OCDS EV chip for the HT45F0005A device.

Pin Description

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

Pin Name	Function	OPT	I/T	O/T	Descriptions
PA0/SCL/SDA/ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SCL	PAS0 IFS	ST	NMOS	I^2C clock line
	SDA	PAS0 IFS	ST	NMOS	I^2C data line
	ICPDA	—	ST	CMOS	ICP data/address
	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
PA1/OPOUT/AN8	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OPOUT	PAS0	—	AN	OPAMP output pin
	AN8	PAS0	AN	—	A/D converter external input 8
PA2/SCL/OPROUT/AN2/ ICPCK/OCDSCK	PA2	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SCL	PAS1 IFS	ST	NMOS	I^2C clock line
	OPROUT	PAS1	—	AN	OPAMP output pin
	AN2	PAS1	AN	—	A/D converter external input 2
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/TC0/CP0P/AN9	PA3	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	TC0	PAS1	ST	—	Timer/Event Counter 0 clock input
	CP0P	PAS1	AN	—	Comparator 0 non-inverting input
	AN9	PAS1	AN	—	A/D converter external input 9

Pin Name	Function	OPT	I/T	O/T	Descriptions
PA4/C2VO/VREF/AN4	PA4	PAPU PAWU PAS2	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	C2VO	PAS2	—	CMOS	Comparator 2 output (before debounce)
	VREF	PAS2	AN	—	A/D converter external reference voltage input
	AN4	PAS2	AN	—	A/D converter external input 4
PA5/CP1N/AN7	PA5	PAPU PAWU PAS2	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CP1N	PAS2	AN	—	Comparator 1 inverting input
	AN7	PAS2	AN	—	A/D converter external input 7
PA6/CP2N/AN6/C5VOD	PA6	PAPU PAWU PAS3	ST	CMOS	eneral purpose I/O. Register enabled pull-up and wake-up
	CP2N	PAS3	AN	—	Comparator 2 inverting input
	AN6	PAS3	AN	—	A/D converter external input 6
	C5VOD	PAS3	—	CMOS	Comparator 5 output (after debounce)
PA7/CP2P/AN5/C5VO	PA7	PAPU PAWU PAS3	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CP2P	PAS3	AN	—	Comparator 2 non-inverting input
	AN5	PAS3	AN	—	A/D converter external input 5
	C5VO	PAS3	—	CMOS	Comparator 5 output (before debounce)
PB0/SDA/OPINN0/OPINP	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SDA	PBS0 IFS	ST	NMOS	I^2C data line
	OPINN0	PBS0	AN	—	OPAMP inverting input 0
	OPINP	PBS0	AN	—	OPAMP non-inverting input
PB1/TC1/SDA/PCK/C1VO/ C3VO/AN12/CP5N	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TC1	PBS0 IFS	ST	—	Timer/Event Counter 1 clock input
	SDA	PBS0 IFS	ST	NMOS	I^2C data line
	PCK	PBS0	—	CMOS	PCK output
	C1VO	PBS0	—	CMOS	Comparator 1 output (before debounce)
	C3VO	PBS0	—	CMOS	Comparator 3 output (before debounce)
	AN12	PBS0	AN	—	A/D converter external input 12
	CP5N	PBS0	AN	—	Comparator 5 inverting input
PB2/CP4P/AN1	PB2	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CP4P	PBS1	AN	—	Comparator 4 non-inverting input
	AN1	PBS1	AN	—	A/D converter external input 1
PB3/PPGIN/CP0N/AN0	PB3	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PPGIN	PBS1	ST	—	PPG external trigger input
	CP0N	PBS1	AN	—	Comparator 0 inverting input
	AN0	PBS1	AN	—	A/D converter external input 0

Pin Name	Function	OPT	I/T	O/T	Descriptions
PB4/SCL/PWMO/C0VO/ OPINN1/AN3	PB4	PBPU PBS2	ST	CMOS	General purpose I/O. Register enabled pull-up
	SCL	PBS2 IFS	ST	NMOS	I^2C clock line
	PWMO	PBS2	—	CMOS	PWM output
	C0VO	PBS2	—	CMOS	Comparator 0 output (before debounce)
	OPINN1	PBS2	AN	—	OPAMP inverting input 1
	AN3	PBS2	AN	—	A/D converter external input 3
PC0/C0VOD/CP3N/AN10	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	C0VOD	PCS0	—	CMOS	Comparator 0 output (after debounce)
	CP3N	PCS0	AN	—	Comparator 3 inverting input
	AN10	PCS0	AN	—	A/D converter external input 10
PC1/C1VOD/PCK/CP3N/SCL/ AN13	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	C1VOD	PCS0	—	CMOS	Comparator 1 output (after debounce)
	PCK	PCS0	—	CMOS	PCK output
	CP3N	PCS0	AN	—	Comparator 3 inverting input
	SCL	PCS0 IFS	ST	NMOS	I^2C clock line
	AN13	PCS0	AN	—	A/D converter external input 3
PC2/LVDIN/C4VO/C2VOD/ SDA/CP5N	PC2	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	LVDIN	PCS1	AN	—	LVD external input
	C4VO	PCS1	—	CMOS	Comparator 4 output (before debounce)
	C2VOD	PCS1	—	CMOS	Comparator 2 output (after debounce)
	SDA	PCS1 IFS	ST	NMOS	I^2C data line
	CP5N	PCS1	AN	—	Comparator 5 inverting input
PC3/TC1/C3VOD/C4VOD/AN11	PC3	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	TC1	PCS1 IFS	ST	—	Timer/Event Counter 1 clock input
	C3VOD	PCS1	—	CMOS	Comparator 3 output (after debounce)
	C4VOD	PCS1	—	CMOS	Comparator 4 output (after debounce)
	AN11	PCS1	AN	—	A/D converter external input 11
PPG	PPG	—	—	PMOS/ NMOS/ CMOS	Programmable pulse generator output pin The PPG pin output active level can be selected by software, either active high for PMOS output, active low for NMOS output or a forced high or low level for CMOS output
VDD	VDD	—	PWR	—	Digital positive power supply
VSS	VSS	—	PWR	—	Digital negative power supply

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

Absolute Maximum Ratings

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

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

Storage Temperature..... -60°C to 150°C

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

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

I_{OL} Total 80mA

Total Power Dissipation 500mW

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

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=25°C

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

Operating Current Characteristics

Ta=25°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	5V	$f_{SYS}=f_{LIRC}$=32kHz	—	30	50	μA
	FAST Mode – HIRC	5V	$f_{SYS}=f_{HIRC}$=16MHz	—	2.5	5.0	mA

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

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB}	SLEEP Mode	3.3V	WDT on	—	1.5	3.0	μA
		5V		—	3	5	μA
	IDLE0 Mode – LIRC	3.3V	f_{SUB} on	—	3	5	μA
		5V		—	5	10	μA
	IDLE1 Mode – HIRC	3.3V	f_{SUB} on, f_{SYS}=16MHz	—	0.80	1.20	mA
		5V		—	1.4	2.0	mA

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.

High Speed Internal 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 5V.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	16MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	16	+1%	MHz
			-40°C~85°C	-2%	16	+2%	
		4.0V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C~85°C	-3.0%	16	+3.0%	
		3.3V~5.5V	25°C	-4.0%	16	+4.0%	
			-40°C~85°C	-6.0%	16	+6.0%	

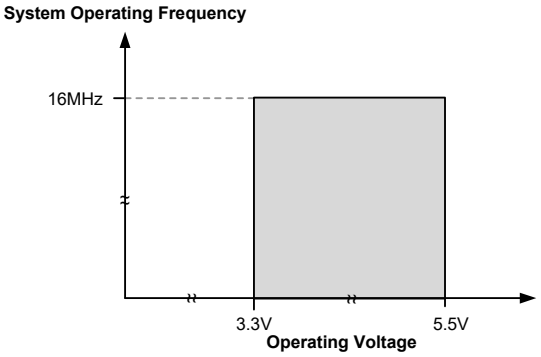
Note: 1. The 5V values for V_{DD} are provided as this is the fixed voltage at which the HIRC frequency is trimmed by the writer.

2. The row below the 5V trim voltage row is provided to show the values for the specific V_{DD} range operating voltage.

Low Speed Internal Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	5V	25°C	-2%	32	+2%	kHz
		3.3V~5.5V	-40°C~85°C	-15%	32	+15%	

Operating Frequency Characteristic Curve



System Start Up Time Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time (Wake-up from Condition where f_{SYS} is off)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	System Start-up Time (Wake-up from Condition where f_{SYS} is on)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	System Speed Switch Time (FAST to SLOW Mode or SLOW to FAST Mode)	—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
t_{RSTD}	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR_{POR}=5V/ms	14	16	18	ms
	System Reset Delay Time (LVRC/WDTC Register Software Reset)	—	—				
	System Reset Delay Time (Reset Source from WDT Overflow Reset)	—	—	14	16	18	ms
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

Note: 1. For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.
2. The time units, shown by the symbols t_{HIRC}, t_{SYS} etc. are the inverse of the corresponding frequency values as provided in the table above. For example $t_{HIRC}=1/f_{HIRC}$, $t_{SYS}=1/f_{SYS}$ etc.
3. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

Ta=25°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	5V	$V_{OL}=0.1V_{DD}$	32	65	—	mA
I_{OH}	Source Current for I/O Ports	5V	$V_{OH}=0.9V_{DD}$	-8	-16	—	mA
R_{PH}	Pull-High Resistance for I/O Ports (Note)	5V	—	10	30	50	kΩ

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{LEAK}	Input Leakage Current	5V	V_{IN}=V_{DD} or V_{IN}=V_{SS}	—	—	±1	μA
t_{TC}	TCn Input Pin Minimum Pulse Width	—	—	25	—	—	ns

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

Memory Electrical Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{RW}	V_{DD} for Read / Write	—	—	V_{DDmin}	—	V_{DDmax}	V
Flash Program Memory							
t_{FWR}	Write Time	—	—	1.364	1.500	1.667	ms
t_{FER}	Erase Time	—	—	2.273	2.500	2.778	ms
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	Data Retention Time	—	Ta=25°C	—	40	—	Year
t_{ACTV}	ROM Activation Time – Wake-up from Power Down Mode	—	—	32	—	64	μs
Data EEPROM Memory							
t_{EERD}	Read Time	—	—	—	—	4	t_{SYS}
t_{EEWR}	Write Time – Byte Mode	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	
	Write Time – Page Mode	—	EWERTS=0	—	2.2	2.7	
		—	EWERTS=1	—	3.0	3.6	
t_{EEER}	Erase Time	—	EWERTS=0	—	3.2	3.9	ms
		—	EWERTS=1	—	3.7	4.5	
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

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

LVD & LVR Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	-5%	2.1	+5%	V
		—	LVR enable, voltage select 2.55V		2.55		
		—	LVR enable, voltage select 3.15V		3.15		
		—	LVR enable, voltage select 3.8V		3.8		

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select LVDIN pin=1.23V	-10%	1.23	+10%	V
		—	LVD enable, voltage select 2.2V	-5%	2.2	+5%	
		—	LVD enable, voltage select 2.4V		2.4		
		—	LVD enable, voltage select 2.7V		2.7		
		—	LVD enable, voltage select 3.0V		3.0		
		—	LVD enable, voltage select 3.3V		3.3		
		—	LVD enable, voltage select 3.6V		3.6		
		—	LVD enable, voltage select 4.0V		4.0		
$I_{LVRLVDBG}$	Operating Current	5V	LVD enable, LVR enable, VBGEN=0	—	20	25	μA
		5V	LVD enable, LVR enable, VBGEN=1	—	180	200	μA
t_{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	18	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	TLVR[1:0]=00B	120	240	480	μs
			TLVR[1:0]=01B	0.5	1.0	2.0	ms
			TLVR[1:0]=10B	1	2	4	
			TLVR[1:0]=11B	2	4	8	
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	TLVD[1:0]=00B/11B	60	140	220	μs
			TLVD[1:0]=01B	90	200	340	
			TLVD[1:0]=10B	150	320	580	

Note: If V_{LVD}=1.23V, it is used to detect the LVDIN pin input voltage. Other V_{LVD} choices are used to detect the power supply V_{DD}.

Reference Voltage Characteristics

Ta=25°C, unless otherwise specified.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{BG}	Bandgap Reference Voltage	—	Ta=-40°C~85°C	-5%	1.23	+5%	V

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

A/D Converter Electrical Characteristics

Ta=25°C, unless otherwise specified.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	2	—	V_{DD}	V
N_R	Resolution	—	—	—	—	12	Bit
DNL	Differential Non-linearity	5V	V_{REF}=V_{DD}, t_{ADCK}=0.5μs Ta=-40°C~85°C	-3	—	+3	LSB
INL	Integral Non-linearity	5V	V_{REF}=V_{DD}, t_{ADCK}=0.5μs Ta=-40°C~85°C	-4	—	+4	LSB
I_{ADC}	Additional Current Consumption for A/D Converter Enable	5V	No load, t_{ADCK}=0.5μs	—	850	1000	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{ADCK}	Clock Period	3.3V~5.5V	—	0.5	—	10.0	μs
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t_{ADS}	Sampling Time	—	—	—	4	—	t_{ADCK}
t_{ADC}	Conversion Time (Including A/D Sample and Hold Time)	—	—	—	16	—	t_{ADCK}

Comparator Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{CMP}	Additional Current for Comparator Enable	5V	CnEN=1, (CMPn Enable, n=0)	—	55	120	μA
		5V	CnEN=1, (CMPn & DACn Enable, n=1~5)	—	500	750	μA
V_{OS}	Input Offset Voltage	—	Without calibration, (CnCOF[5:0]=100000B, n=0)	-15	—	15	mV
		—	Without calibration, (CnCOF[4:0]=10000B, n=1~5)	-15	—	15	mV
		—	With calibration	-2	—	2	mV
V_{CM}	Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}-1.0	V
V_{HYS}	CMPn Hysteresis (n=0~5)	5V	CnHYSON[1:0]=00B	0	0	5	mV
			CnHYSON[1:0]=01B	15	30	50	
			CnHYSON[1:0]=10B	30	60	90	
			CnHYSON[1:0]=11B	40	90	130	
t_{RP}	Response Time	5V	With 10mV overdrive	—	—	2	μs
			With 60mV overdrive	—	—	1.5	μs
	CMPn Response Time (n=1~5)	5V	CnDA=1100 0111B, DAC V_{REF}=V_{DD}, CMPn Input=0V~5.0V (Note)	—	0.5	1.0	μs
			CnDA=00000101B, DAC V_{REF}=V_{DD}, CMPn Input=0V~5.0V (Note)	—	0.5	1.0	μs

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{CIRP}	CMP0 Interrupt Response Time	—	Hysteresis disable, debounce disable	—	0.8	1.5	μs
			Hysteresis disable, debounce enable, C0DBC[5:0]=000001B~101111B	—	(1~47) ×t_{DBDL} +0.8	(1~47) ×t_{DBDL} +1.5	μs
			Hysteresis disable, debounce enable, C0DBC[5:0]=110000B~111111B	—	48×t_{DBDL} +0.8	48×t_{DBDL} +1.5	μs
	CMPn Interrupt Response Time (n=1~3, 5)	—	CnHYSON[1:0]=00B, CnDB[2:0]=000B	—	0.8	1.5	μs
			CnHYSON[1:0]=00B, CnDB[2:0]=001B	—	3×t_{DBDL} +0.8	4×t_{DBDL} +1.5	
			CnHYSON[1:0]=00B, CnDB[2:0]=010B	—	7×t_{DBDL} +0.8	8×t_{DBDL} +1.5	
			CnHYSON[1:0]=00B, CnDB[2:0]=011B	—	15×t_{DBDL} +0.8	16×t_{DBDL} +1.5	
			CnHYSON[1:0]=00B, CnDB[2:0]=100B	—	31×t_{DBDL} +0.8	32×t_{DBDL} +1.5	
			CnHYSON[1:0]=00B, CnDB[2:0]=101B	—	63×t_{DBDL} +0.8	64×t_{DBDL} +1.5	
			CnHYSON[1:0]=00B, CnDB[2:0]=110B	—	127×t_{DBDL} +0.8	128×t_{DBDL} +1.5	
			CnHYSON[1:0]=00B, CnDB[2:0]=000B	—	255×t_{DBDL} +0.8	256×t_{DBDL} +1.5	
t_{CIRP}	CMP4 Interrupt Response Time	—	C4HYSON[1:0]=00B, C4DB[2:0]=000B	—	0.8	1.5	μs
			C4HYSON[1:0]=00B, C4DB[2:0]=001B	—	3×t_{DEB} +0.8	4×t_{DEB} +1.5	
			C4HYSON[1:0]=00B, C4DB[2:0]=010B	—	7×t_{DEB} +0.8	8×t_{DEB} +1.5	
			C4HYSON[1:0]=00B, C4DB[2:0]=011B	—	15×t_{DEB} +0.8	16×t_{DEB} +1.5	
			C4HYSON[1:0]=00B, C4DB[2:0]=100B	—	31×t_{DEB} +0.8	32×t_{DEB} +1.5	
			C4HYSON[1:0]=00B, C4DB[2:0]=101B	—	63×t_{DEB} +0.8	64×t_{DEB} +1.5	
			C4HYSON[1:0]=00B, C4DB[2:0]=110B	—	127×t_{DEB} +0.8	128×t_{DEB} +1.5	
			C4HYSON[1:0]=00B, C4DB[2:0]=111B	—	255×t_{DEB} +0.8	256×t_{DEB} +1.5	
t_{INTDY}	INT00 Delay Time (Include Debounce Time)	5V	C0DLY[5:0]=000000B~101111B, C0DBC[5:0]=000000B	Typ. -0.2	(0~47) ×t_{DBDL} +0.08	Typ. +0.2	μs
		5V	C0DLY[5:0]=110000B~111111B, C0DBC[5:0]=000000B	Typ. -0.2	48×t_{DBDL} +0.08	Typ. +0.2	μs
DNL	Differential Non-linearity	5V	DAC V_{REF}=V_{DD}	-1	—	+1	LSB
INL	Integral Non-linearity	5V	DAC V_{REF}=V_{DD}	-1.5	—	+1.5	LSB
t_{PPGIN}	External PPGIN Input Minimum Pulse Width	—	—	0.1	—	—	μs

Note: As the CMPn (n=1~5) has one end connected to a D/A converter, it is necessary to pay attention to whether it conforms to the comparator input common mode range, to ensure the comparator normal operation.

Operational Amplifier Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OPA}	Additional Current for Operational Amplifier Enable	5V	No load	—	300	600	μA
V_{OS}	Input Offset Voltage	5V	Without calibration (OPOOF[5:0]=100000B)	-15	—	15	mV
		5V	With calibration	-2	—	2	
V_{CM}	Common Mode Voltage Range	5V	—	V_{SS}	—	V_{DD}-1.4	V
V_{OR}	Maximum Output Voltage Range	5V	—	V_{SS}+0.1	—	V_{DD}-0.1	V
R_{OPAR3}	OPAR3 Resistance	5V	—	0.75	1.00	1.25	kΩ
R_{OPAR4}	OPAR4 Resistance	5V	—	7.5	10.0	12.5	kΩ
SR	Slew Rate	5V	No load	0.6	1.8	—	V/μs
GBW	Gain Bandwidth	5V	R_{LOAD}=1MΩ, C_{LOAD}=100pF	—	2200	—	kHz
PSRR	Power Supply Rejection Ratio	5V	—	60	80	—	dB
CMRR	Common Mode Rejection Ratio	5V	—	60	80	—	dB
Ga	PGA Gain Accuracy (Note)	5V	Relative gain	-5	—	5	%

Note: The PGA gain accuracy is guaranteed only when the PGA output voltage meets the V_{OR} specification.

Programmable Pulse Generator Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{PPGF}	PPG Pin Floating Voltage (Note)	—	LVR enable, voltage select 2.1V	-5%	2.1	+5%	V
		—	LVR enable, voltage select 2.55V		2.55		V
		—	LVR enable, voltage select 3.15V		3.15		V
		—	LVR enable, voltage select 3.8V		3.8		V
I_{OL}	Sink Current for PPG	5V	V_{OL}=0.1V_{DD}	32	65	—	mA
I_{OH}	Source Current for PPG	5V	V_{OH}=0.9V_{DD}	-8	-16	—	mA

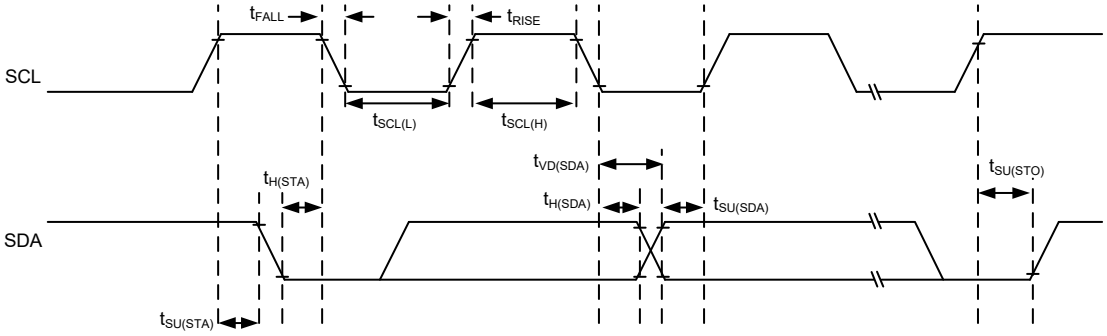
Note: When V_{DD}<V_{LVR}, the PPG pin is floating.

I²C Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{I2C}	I²C Standard Mode (100kHz) f_{SYS} Frequency (Note)	—	No clock debounce	2	—	—	MHz
			2 system clock debounce	4	—	—	
			4 system clock debounce	4	—	—	
	I²C Fast Mode (400kHz) f_{SYS} Frequency (Note)	—	No clock debounce	4	—	—	MHz
			2 system clock debounce	8	—	—	
			4 system clock debounce	8	—	—	
f_{SCL}	SCL Clock Frequency	3V/5V	Standard mode	—	—	100	kHz
			Fast mode	—	—	400	
$t_{SCL(H)}$	SCL Clock High Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
$t_{SCL(L)}$	SCL Clock Low Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
t_{FALL}	SCL and SDA Fall Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
t_{RISE}	SCL and SDA Rise Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
$t_{SU(SDA)}$	SDA Data Setup Time	3V/5V	Standard mode	0.25	—	—	μs
			Fast mode	0.1	—	—	
$t_{H(SDA)}$	SDA Data Hold Time	3V/5V	—	0.1	—	—	μs
$t_{VD(SDA)}$	SDA Data Valid Time	3V/5V	—	—	—	0.6	μs
$t_{SU(STA)}$	Start Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	
$t_{H(STA)}$	Start Condition Hold Time	3V/5V	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.

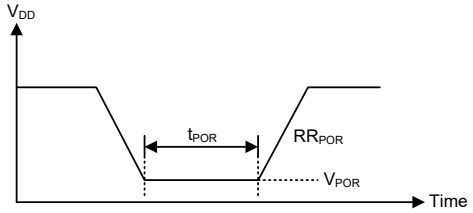


I²C Timing Diagram

Power-on Reset Characteristics

Ta=25°C

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



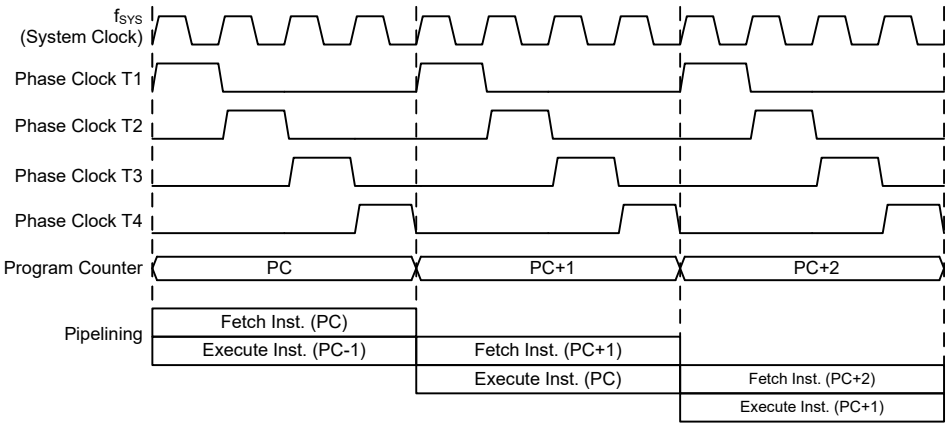
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one or two cycles for most of the standard or extended instructions respectively, with the exception of branch or call instructions which 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 low-cost, high-volume production for controller applications.

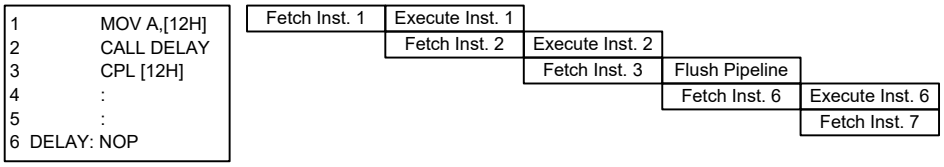
Clocking and Pipelining

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



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. 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)**
PC11~PC8	PCL7~PCL0

Program Counter

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

Program Counter Read Registers

The Program Counter Read registers are a read only register pair for reading the program counter value which indicates the current program execution address. Read the low byte register first then

the high byte register. Reading the low byte register, PCRL, will read the low byte data of the current program execution address, and place the high byte data of the program counter into the 8-bit PCRH buffer. Then reading the PCRH register will read the corresponding data from the 8-bit PCRH buffer.

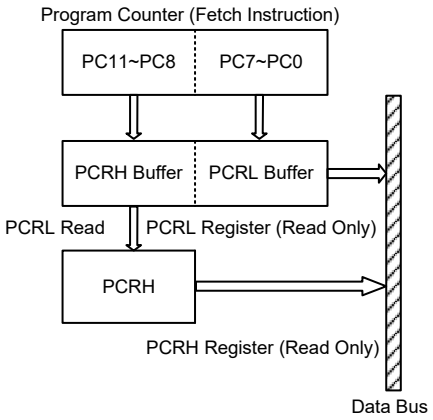
The following example shows how to read the current program execution address. When the current program execution address is 123H, the steps to execute the instructions are as follows:

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

MOV A, PCRH → the ACC value is 01H.

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

LMOV A, PCRH → the ACC value is 01H.



- **PCRL Register**

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

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

- **PCRH Register**

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

Bit 7~4 Unimplemented, read as "0"

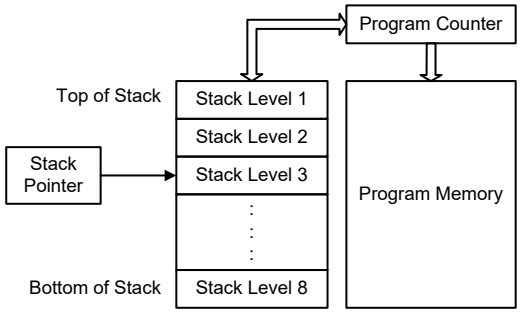
Bit 3~0 **D11~D8**: Program Counter Read High byte register bit 3 ~ bit 0

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 8 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[2:0]. 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	—	—	—	—	D2	D1	D0
R/W	R/W	—	—	—	—	R	R	R
POR	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 6~3 Unimplemented, read as "0"

Bit 2~0 **D2~D0**: Stack pointer register bit 2 ~ 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 9 times continuously and the RET instruction is not executed during the period, the corresponding changes of the STKPTR[2:0] and OSF bits are as follows:

CALL Execution Times	0	1	2	3	4	5	6	7	8	9
STKPTR[2:0] Bit Value	0	1	2	3	4	5	6	7	0	1
OSF Bit Value	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 8 times continuously, the corresponding changes of the STKPTR[2:0] and OSF bits are as follows:

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

Arithmetic and Logic Unit – ALU

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

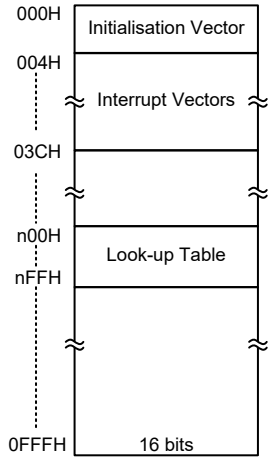
- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
  LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,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

Flash Program Memory

The Program Memory is the location where the user code or program is stored. For the device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing users 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 4K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

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

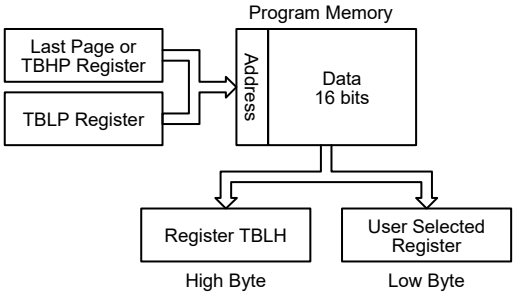


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "0F00H " which refers to the start address of the last page within the 4K 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 "0F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by the TBHP and TBLP registers if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

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

Table Read Program Example

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

In Circuit Programming – ICP

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

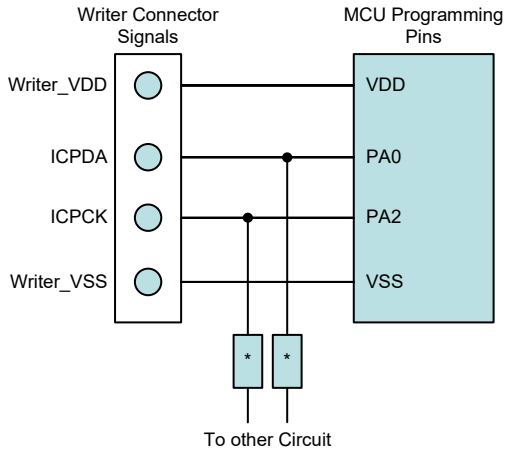
As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or 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 Holtek Flash MCU to Writer Programming Pin correspondence table is as follows:

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

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

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



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

On-Chip Debug Support – OCDS

There is an EV chip named HT45V0005A which is used to emulate the HT45F0005A device. The EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

Data Memory

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

Categorized into two types, the first of these is an area of RAM, 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.

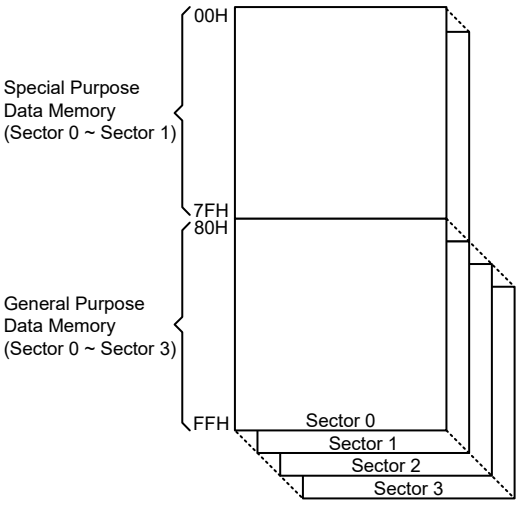
Structure

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

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

Special Purpose Data Memory	General Purpose Data Memory	
Located Sectors	Capacity	Sector: Address
0, 1	512×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH

Data Memory Summary



Data Memory Structure

Data Memory Addressing

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

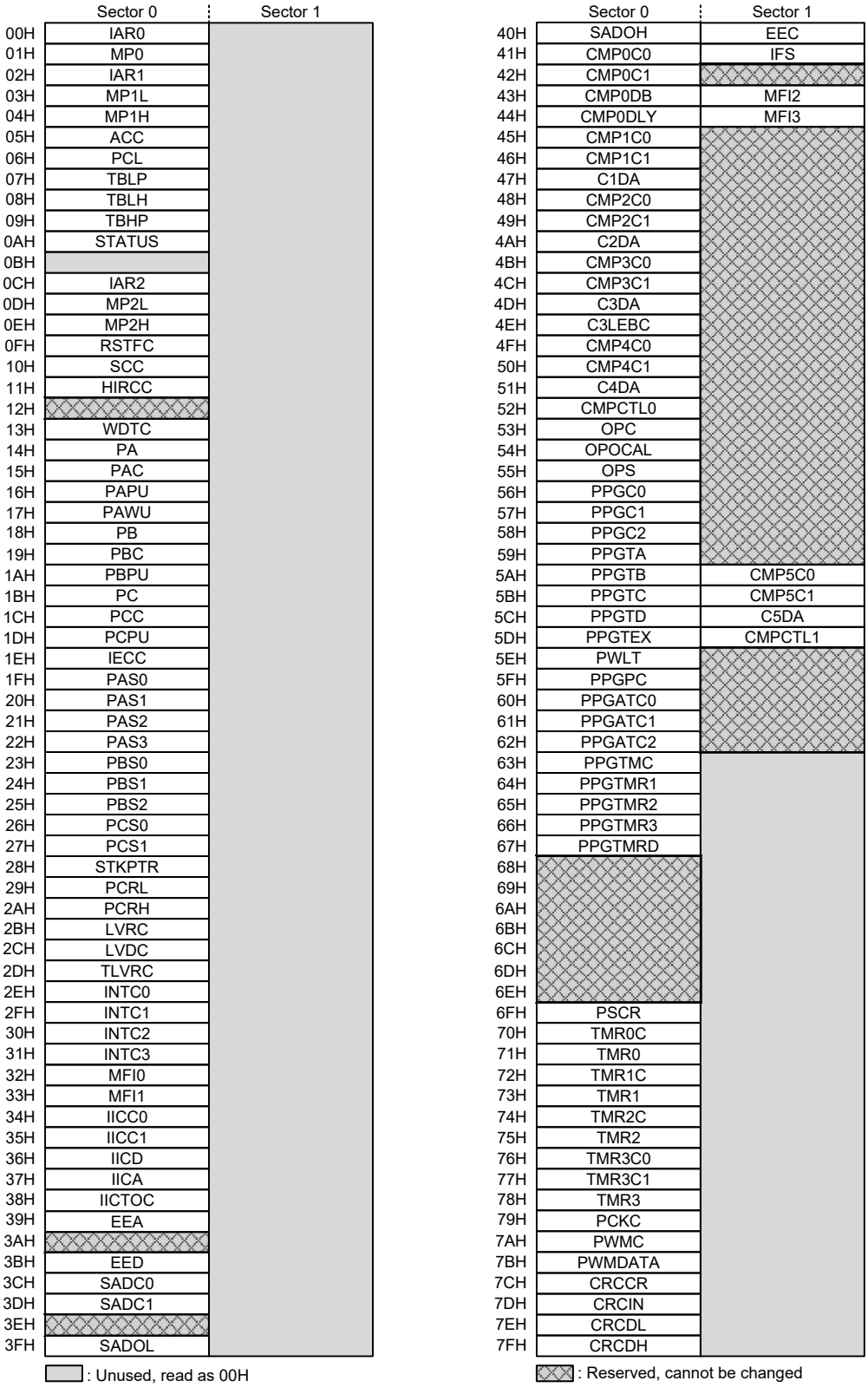
Direct Addressing can be used in all sectors using the extended instructions which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address “m” in the extended instructions has 10 valid bits for this device, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user programing 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”.



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 the MP1L/MP1H register pair and IAR2 register together with the MP2L/MP2H register pair can access data from any Data Memory Sector. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers will return a result of "00H" and writing to the registers will result in no operation.

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

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L, MP2H, are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Sector 0, while MP1L/MP1H together with IAR1 and MP2L/MP2H together with IAR2 are used to access data from all sectors according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all sectors using the extended 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                 ; increase 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 examples shown above, no reference is made to specific Data Memory addresses.

Direct Addressing Program Example Using Extended Instructions

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

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

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

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

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

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

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

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

• **STATUS Register**

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

"x": Unknown

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

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

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

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

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

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

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

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

EEPROM Data Memory

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 128×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and write operations to the EEPROM are carried out in either the byte mode or page mode determined by the mode selection bit, MODE, in the control register, EEC.

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As the EEA 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
EEA	D7	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD

EEPROM Register List

• EEA Register

Bit	7	6	5	4	3	2	1	0
Name	D7	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **D7**: Reserved, must be fixed at "0"

Bit 6~0 **EEA6~EEA0**: Data EEPROM address register bit 6 ~ bit 0
Data EEPROM address bit 6 ~ bit 0

• EED Register

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

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

• EEC Register

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

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

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

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

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

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

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

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

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

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

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

Page Read Mode

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

The EEPROM address higher 3 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 3-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 3 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 3-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 EEA register and the dummy data to be written should be placed in the EED register. The maximum data length for a page is 16 bytes. Note that the write operation to the EED register is used to tag address, it must be implemented to determine which addresses to be erased. When the page dummy data is completely written, then the EREN bit in the EEC register should be set high to enable erase operations and the ER bit must be immediately set high to initiate the EEPROM erase process. These two instructions must be executed in two consecutive instruction cycles to activate an erase operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing an erase operation and then set again after a valid erase activation procedure has completed.

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

Write Operation to the EEPROM

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

Byte Write Mode

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

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

Page Write Mode

Before a page write operation is executed, it is important to ensure that a relevant page erase operation has been successfully executed. The EEPROM page write operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page write. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM

write enable control bit, namely WREN, is changed from "1" to "0", the internal page buffer will also be cleared. Note that when the WREN bit is changed from "0" to "1", the internal page buffer will not be cleared. A page write is initiated in the same way as a byte write initiation except that the EEPROM data can be written up to 16 bytes. The EEPROM address higher 3 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 3-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 EEA register and the data to be written should be placed in the EED register. The maximum data length for a page is 16 bytes. Note that when a data byte is written into the EED register, then the data in the EED register will be loaded into the internal page buffer and the current address value will automatically be incremented by one. When the page data is completely written into the page buffer, then the WREN bit in the EEC register should be set high to enable write operations and the WR bit must be immediately set high to initiate the EEPROM write process. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

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

Write Protection

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

EEPROM Interrupt

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

Programming Considerations

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

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

Programming Examples

Reading a Data Byte from the EEPROM – polling method

```
MOV A, 040H              ; setup memory pointer low byte MP1L
MOV MP1L, A              ; MP1 points to EEC register
MOV A, 01H               ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4               ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
SET IAR1.1               ; set RDEN bit, enable read operations
SET IAR1.0               ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0               ; check for read cycle end
JMP BACK
CLR IAR1                 ; disable EEPROM read function
CLR MP1H
MOV A, EED               ; move read data to register
MOV READ_DATA, A
```

Reading a Data Page from the EEPROM – polling method

```
MOV A, 040H              ; setup memory pointer low byte MP1L
MOV MP1L, A              ; MP1 points to EEC register
MOV A, 01H               ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4               ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
SET IAR1.1               ; set RDEN bit, enable read operations
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL READ
CALL READ
:
:
JMP PAGE_READ_FINISH
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
READ:
SET IAR1.0               ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0               ; check for read cycle end
```

```
JMP BACK
MOV A, EED                ; move read data to register
MOV READ_DATA, A
RET
:
PAGE_READ_FINISH:
CLR IAR1                  ; disable EEPROM read function
CLR MP1H
```

Erasing a Data Page to the EEPROM − polling method

```
MOV A, 040H               ; setup memory pointer low byte MP1L
MOV MP1L, A               ; MP1 points to EEC register
MOV A, 01H                ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4                ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP Erase_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA        ; user defined data, erase mode don't care data value
MOV EED, A
RET
:
Erase_START:
CLR EMI
SET IAR1.6                ; set EREN bit, enable erase operations
SET IAR1.5                ; start Erase Cycle - set ER bit - executed immediately
                          ; after setting EREN bit
SET EMI
BACK:
SZ  IAR1.5                ; check for erase cycle end
JMP BACK
CLR MP1H
```

Writing a Data Byte to the EEPROM – polling method

```
MOV A, 040H               ; setup memory pointer low byte MP1L
MOV MP1L, A               ; MP1 points to EEC register
MOV A, 01H                ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4                ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit - executed immediately
                          ; after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2                ; check for write cycle end
JMP BACK
CLR MP1H
```

Writing a Data Page to the EEPROM – polling method

```
MOV A, 040H               ; setup memory pointer low byte MP1L
MOV MP1L, A               ; MP1 points to EEC register
MOV A, 01H                ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4                ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP WRITE_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
RET
:
WRITE_START:
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit - executed immediately
                          ; after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2                ; check for write cycle end
JMP BACK
CLR MP1H
```

Oscillators

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

Oscillator Overview

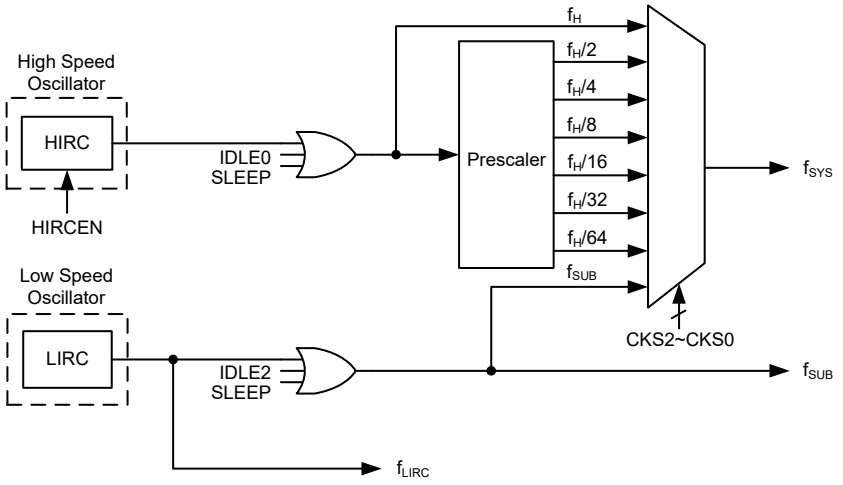
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer. Fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillators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

Type	Name	Frequency
Internal High Speed RC	HIRC	16MHz
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 speed oscillator is the internal 16MHz 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 a fixed frequency of 16MHz. 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.

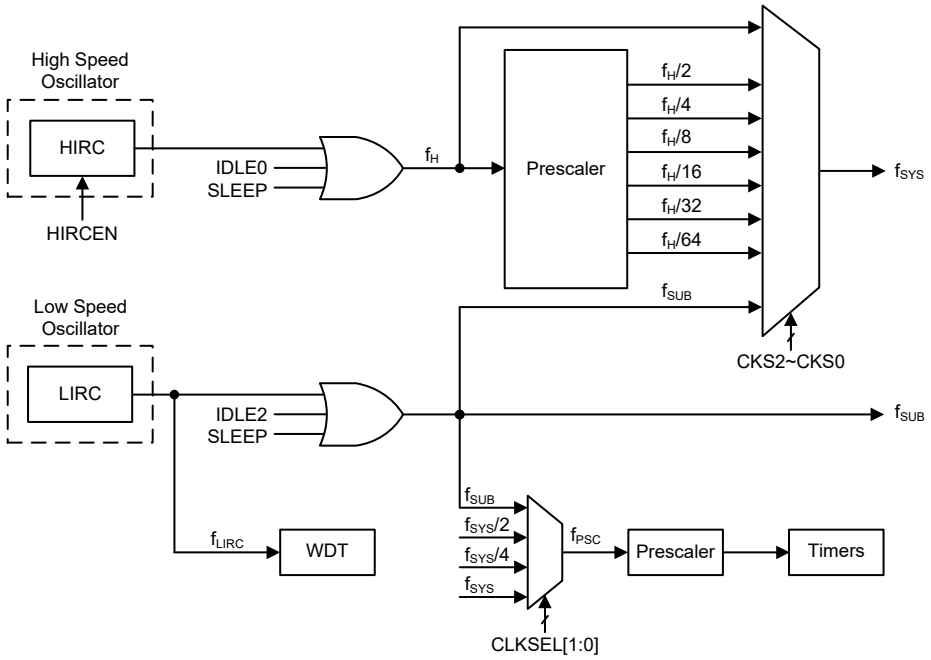
Operating Modes and System Clocks

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, users can optimise the operation of their microcontroller to achieve the best performance/power ratio.

System Clocks

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

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



Device Clock Configurations

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

System Operation Modes

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

Operation Mode	CPU	Register Setting			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
FAST	On	x	x	000~110	f_H~$f_H/64$	On	On	On
SLOW	On	x	x	111	f_{SUB}	On/Off (1)	On	On
IDLE0	Off	0	1	000~110	Off	Off	On	On
				111	On			
IDLE1	Off	1	1	xxx	On	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off	On
				111	Off			
SLEEP	Off	0	0	xxx	Off	Off	Off	On (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 since the WDT function is always enabled even 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 the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source from the HIRC high speed oscillator. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from the LIRC oscillator.

SLEEP Mode

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

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
HIRCC	—	—	—	—	—	—	HIRCF	HIRCEN

System Operating Mode Control Register List

• **SCC Register**

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

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

Bit 4~2 Unimplemented, read as 0

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

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

Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.

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

• **HIRCC Register**

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

Bit 7~2 Unimplemented, read as "0"

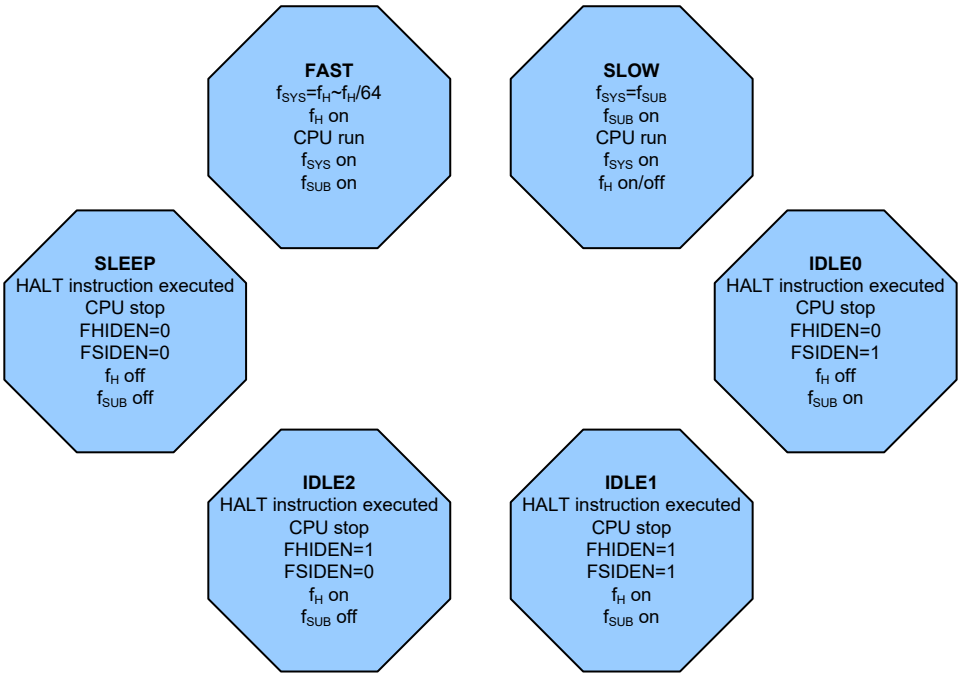
Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set high to enable the HIRC oscillator, the HIRCF bit will first be cleared to zero and then set high after the HIRC oscillator is stable.

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

Operating Mode Switching

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

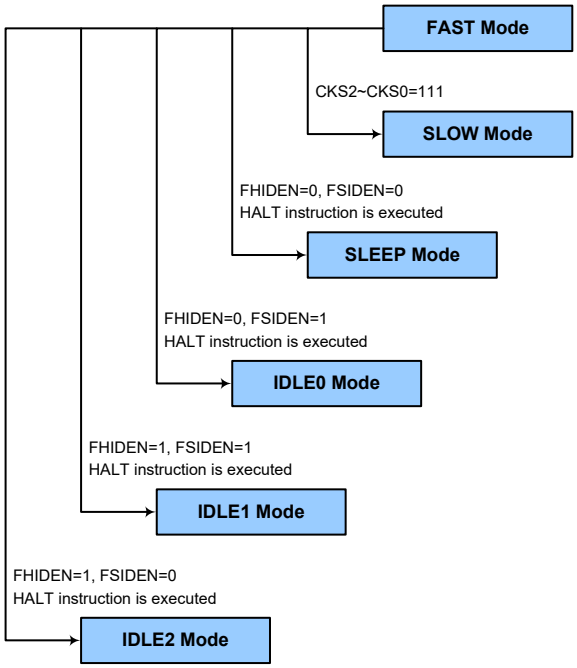
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Mode to the SLEEP/IDLE Mode is executed via the HALT instruction. When a HALT instruction is executed, whether the device enter the IDLE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

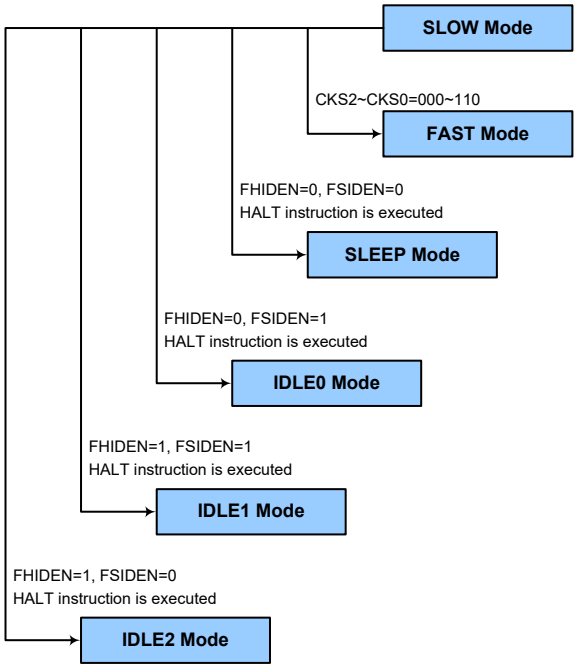
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



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". In this mode all the clocks and functions will be switched off except the WDT function. When this instruction is executed under the conditions described above, the following will occur:

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

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

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

Entering the IDLE2 Mode

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

- The f_H clock will be on but the f_{SUB} clock will be off and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.

- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting as the WDT function is always enabled.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps in the SLEEP and IDLE0 modes, 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.

In the IDLE1 and IDLE2 modes 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, the PDF flag will be set high. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction. If the system is woken up by a WDT overflow, a Watchdog Timer reset will be initiated and the TO flag will be set high. The TO flag is set if a WDT time-out occurs and causes a wake-up that only resets the Program Counter and Stack Pointer, other flags remain in their original status.

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

Watchdog Timer

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

Watchdog Timer Clock Source

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

Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period as well as the WDT enable and MCU software reset operations.

• WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT function software control
10101 or 01010: Enable
Other values: Reset MCU
When these bits are changed 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	—	—	—	—	—	LVRF	LRF	WRF
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	x	0	0

"x": Unknown

Bit 7~3 Unimplemented, read as "0"

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

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

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

Watchdog Timer Operation

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

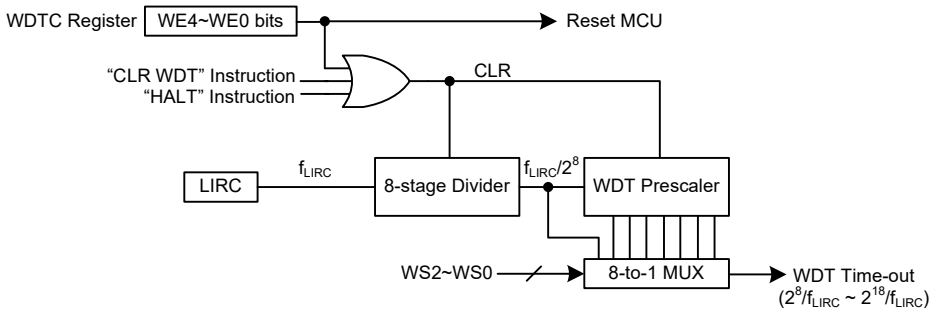
WE4~WE0 Bits	WDT Function
01010B or 10101B	Enable
Any other values	Reset MCU

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

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

The maximum time-out period is when the 2^{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 time-out 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.

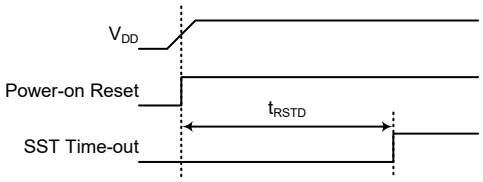
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

Reset Functions

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

Power-on Reset

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

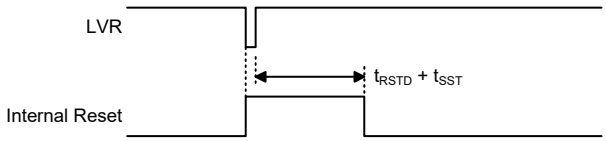


Power-on Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled in the FAST or SLOW 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, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVD & LVR Electrical Characteristics. If the low 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 some different 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 high. 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 IDLE or SLEEP mode.



Low Voltage Reset Timing Chart

• Low Voltage Reset Registers

The LVRC and TLVRC registers are used to control the Low Voltage Reset function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
LVRC	LVS7	LVS6	LVS5	LVS4	LVS3	LVS2	LVS1	LVS0
TLVRC	—	—	—	—	—	—	TLVR1	TLVR0

Low Voltage Reset Register List

• 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 control
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Any other value: Generates MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by the defined LVR voltage values above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps more than a t_{LVR} time. The actual t_{LVR} value can be selected by the TLVR1~TLVR0 bits in the TLVRC register. In this situation the register contents will remain the same after such a reset occurs.

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

• TLVRC Register

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

Bit 7~2 Unimplemented, read as "0"

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

• **RSTFC Register**

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

"x": Unknown

Bit 7~3 Unimplemented, read as "0"

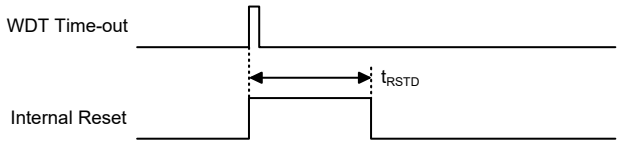
Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set high when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

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

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

Watchdog Time-out Reset during Normal Operation

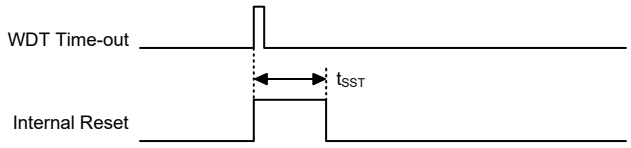
When the Watchdog time-out Reset during normal operation in the FAST or SLOW Mode occurs, the Watchdog time-out flag TO will be set high.



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 zero and the TO and PDF flags will be set high. Refer to the System Start Up Time Characteristics for t_{SST} details.



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

Reset Initial Conditions

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

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

"u": Unchanged

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

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

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

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	---- xxxx	---- uuuu	---- uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu
SCC	001- --00	001- --00	uuu- --uu
HIRCC	---- --01	---- --01	---- --uu
WDTC	0101 0011	0101 0011	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

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PB	---1 1111	---1 1111	---u uuuu
PBC	---1 1111	---1 1111	---u uuuu
PBPU	---0 0000	---0 0000	---u uuuu
PC	---- 1111	---- 1111	---- uuuu
PCC	---- 1111	---- 1111	---- uuuu
PCPU	---- 0000	---- 0000	---- uuuu
IECC	0000 0000	0000 0000	uuuu uuuu
PAS0	-000 -000	-000 -000	-uuu -uuu
PAS1	-000 -000	-000 -000	-uuu -uuu
PAS2	-000 -000	-000 -000	-uuu -uuu
PAS3	-000 -000	-000 -000	-uuu -uuu
PBS0	-000 -000	-000 -000	-uuu -uuu
PBS1	-000 -000	-000 -000	-uuu -uuu
PBS2	---- -000	---- -000	---- -uuu
PCS0	-000 -000	-000 -000	-uuu -uuu
PCS1	-000 -000	-000 -000	-uuu -uuu
STKPTR	0--- -000	0--- -000	u--- -000
PCRL	0000 0000	0000 0000	uuuu uuuu
PCRH	---- 0000	---- 0000	---- uuuu
LVRC	0101 0101	0101 0101	uuuu uuuu
LVDC	0000 0000	0000 0000	uuuu uuuu
TLVRC	---- --01	---- --01	---- --uu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	0000 0000	0000 0000	uuuu uuuu
MFI0	--00 --00	--00 --00	--uu --uu
MFI1	--00 --00	--00 --00	--uu --uu
IICC0	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	0000 000-	0000 000-	uuuu uuu-
IICTOC	0000 0000	0000 0000	uuuu uuuu
EEA	0000 0000	0000 0000	uuuu uuuu
EED	0000 0000	0000 0000	uuuu uuuu
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	uuuu uuuu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
CMP0C0	0010 0000	0010 0000	uuuu uuuu
CMP0C1	0000 0100	0000 0100	uuuu uuuu
CMP0DB	0-00 0000	0-00 0000	u-uu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
CMP0DLY	--00 0000	--00 0000	--uu uuuu
CMP1C0	0001 0000	0001 0000	uuuu uuuu
CMP1C1	0000 0000	0000 0000	uuuu uuuu
C1DA	0000 0000	0000 0000	uuuu uuuu
CMP2C0	0001 0000	0001 0000	uuuu uuuu
CMP2C1	0000 0000	0000 0000	uuuu uuuu
C2DA	0000 0000	0000 0000	uuuu uuuu
CMP3C0	0001 0000	0001 0000	uuuu uuuu
CMP3C1	0000 0000	0000 0000	uuuu uuuu
C3DA	0000 0000	0000 0000	uuuu uuuu
C3LEBC	0000 0000	0000 0000	uuuu uuuu
CMP4C0	0001 0000	0001 0000	uuuu uuuu
CMP4C1	0000 0000	0000 0000	uuuu uuuu
C4DA	0000 0000	0000 0000	uuuu uuuu
CMPCTL0	0000 00-0	0000 00-0	uuuu uu-u
OPC	0--0 --00	0--0 --00	u--u --uu
OPOCAL	0010 0000	0010 0000	uuuu uuuu
OPS	---0 0000	---0 0000	---u uuuu
PPGC0	000- 0000	000- 0000	uuu- uuuu
PPGC1	0000 0000	0000 0000	uuuu uuuu
PPGC2	---0 0000	---0 0000	---u uuuu
PPGTA	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGTB	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGTC	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGTD	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGTEX	-x-x -x-x	-x-x -x-x	-u-u -u-u
PWLT	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGPC	0000 0000	0000 0000	uuuu uuuu
PPGATC0	0000 0000	0000 0000	uuuu uuuu
PPGATC1	0--0 0000	0--0 0000	u--u uuuu
PPGATC2	-000 0000	-000 0000	-uuu uuuu
PPGTMC	---0 --00	---0 --00	---u --uu
PPGTMR1	0000 0000	0000 0000	uuuu uuuu
PPGTMR2	0000 0000	0000 0000	uuuu uuuu
PPGTMR3	0000 0000	0000 0000	uuuu uuuu
PPGTMRD	0000 0000	0000 0000	uuuu uuuu
PSCR	---- -000	---- -000	---- -uuu
TMR0C	0000 1000	0000 1000	uuuu uuuu
TMR0	0000 0000	0000 0000	uuuu uuuu
TMR1C	0000 1000	0000 1000	uuuu uuuu
TMR1	0000 0000	0000 0000	uuuu uuuu
TMR2C	00-0 -000	00-0 -000	uu-u -uuu
TMR2	0000 0000	0000 0000	uuuu uuuu
TMR3C0	0000 1000	0000 1000	uuuu uuuu
TMR3C1	00-- ---0	00-- ---0	uu-- ---u
TMR3	0000 0000	0000 0000	uuuu uuuu
PCKC	-000 0000	-000 0000	-uuu uuuu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PWMC	00-- -000	00-- -000	uu-- -uuu
PWMDATA	0000 0000	0000 0000	uuuu uuuu
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
IFS	-000 0000	-000 0000	-uuu uuuu
MFI2	--00 --00	--00 --00	--uu --uu
MFI3	--00 --00	--00 --00	--uu --uu
CMP5C0	0001 0000	0001 0000	uuuu uuuu
CMP5C1	0000 0000	0000 0000	uuuu uuuu
C5DA	0000 0000	0000 0000	uuuu uuuu
CMPCTL1	---- --00	---- --00	---- --uu

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

Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA~PC. 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	—	—	—	PB4	PB3	PB2	PB1	PB0
PBC	—	—	—	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	—	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	—	—	—	PC3	PC2	PC1	PC0
PCC	—	—	—	—	PCC3	PCC2	PCC1	PCC0
PCPU	—	—	—	—	PCPU3	PCPU2	PCPU1	PCPU0

“—”: Unimplemented, read as “0”

I/O Logic Function Register List

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the PAPU~PCPU 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. Otherwise, the pull-high resistors cannot be enabled.

• **PxPU Register**

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

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

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

Port A Wake-up

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

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

• **PAWU Register**

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

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

I/O Port Control Registers

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

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

• **PxC Register**

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

PxCn: I/O Port x pin type selection
0: Output
1: Input
The PxCn bit is used to control the pin type selection. Here the "x" can be A, B or C. 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 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 TC0, TC1,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 an input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	—	PAS06	PAS05	PAS04	—	PAS02	PAS01	PAS00
PAS1	—	PAS16	PAS15	PAS14	—	PAS12	PAS11	PAS10
PAS2	—	PAS26	PAS25	PAS24	—	PAS22	PAS21	PAS20
PAS3	—	PAS36	PAS35	PAS34	—	PAS32	PAS31	PAS30
PBS0	—	PBS06	PBS05	PBS04	—	PBS02	PBS01	PBS00
PBS1	—	PBS16	PBS15	PBS14	—	PBS12	PBS11	PBS10
PBS2	—	—	—	—	—	PBS22	PBS21	PBS20

Register Name	Bit							
	7	6	5	4	3	2	1	0
PCS0	—	PCS06	PCS05	PCS04	—	PCS02	PCS01	PCS00
PCS1	—	PCS16	PCS15	PCS14	—	PCS12	PCS11	PCS10
IFS	—	D6	D5	IFS4	IFS3	IFS2	IFS1	IFS0

Pin-shared Function Selection Register List

- **PAS0 Register**

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

Bit 7 Unimplemented, read as "0"

Bit 6~4 **PAS06~PAS04**: PA1 pin-shared function selection
000: PA1
001: OPOUT
010: AN8
011: OPOUT & AN8
100~111: PA1

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PAS02~PAS00**: PA0 pin-shared function selection
000: PA0
001: SCL
010: SDA
011: Reserved
100~111: PA0

- **PAS1 Register**

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

Bit 7 Unimplemented, read as "0"

Bit 6~4 **PAS16~PAS14**: PA3 pin-shared function selection
000: PA3/TC0
001: CP0P
010: AN9
011: CP0P & AN9
100~111: PA3/TC0

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PAS12~PAS10**: PA2 pin-shared function selection
000: PA2
001: SCL
010: OPROUT
011: AN2
100: OPROUT & AN2
101: Reserved
110~111: PA2

• **PAS2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PAS26	PAS25	PAS24	—	PAS22	PAS21	PAS20
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~4 **PAS26~PAS24**: PA5 pin-shared function selection
000: PA5
001: Reserved
010: CP1N
011: AN7
100: CP1N & AN7
101~111: PA5

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PAS22~PAS20**: PA4 pin-shared function selection
000: PA4
001: C2VO
010: VREF
011: AN4
100: Reserved
101~111: PA4

• **PAS3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PAS36	PAS35	PAS34	—	PAS32	PAS31	PAS30
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~4 **PAS36~PAS34**: PA7 pin-shared function selection
000: PA7
001: CP2P
010: AN5
011: CP2P & AN5
100: Reserved
101: Reserved
110: C5VO
111: PA7

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PAS32~PAS30**: PA6 pin-shared function selection
000: PA6
001: CP2N
010: AN6
011: CP2N & AN6
100: C5VOD
101~111: PA6

• **PBS0 Register**

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

Bit 7 Unimplemented, read as "0"

Bit 6~4 **PBS06~PBS04**: PB1 pin-shared function selection
000: PB1/TC1
001: SDA
010: PCK
011: C1VO
100: C3VO
101: AN12
110: CP5N
111: PB1/TC1

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PBS02~PBS00**: PB0 pin-shared function selection
000: PB0
001: SDA
010: OPINN0
011: OPINP
100~111: PB0

• PBS1 Register

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

Bit 7 Unimplemented, read as "0"

Bit 6~4 **PBS16~PBS14**: PB3 pin-shared function selection
000: PB3/PPGIN
001: CP0N
010: AN0
011: CP0N & AN0
100~111: PB3/PPGIN

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PBS12~PBS10**: PB2 pin-shared function selection
000: PB2
001: CP4P
010: AN1
011: CP4P & AN1
100: Reserved
101~111: PB2

• PBS2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PBS22	PBS21	PBS20
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **PBS22~PBS20**: PB4 pin-shared function selection
000: PB4
001: SCL
010: PWMO
011: C0VO
100: OPINN1
101: AN3
110~111: PB4

• **PCS0 Register**

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

Bit 7 Unimplemented, read as "0"

Bit 6~4 **PCS06~PCS04**: PC1 pin-shared function selection
000: PC1
001: C1VOD
010: PCK
011: CP3N
100: SCL
101: Reserved
110: Reserved
111: AN13

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PCS02~PCS00**: PC0 pin-shared function selection
000: PC0
001: C0VOD
010: CP3N
011: AN10
100: CP3N & AN10
101: Reserved
110: Reserved
111: PC0

• **PCS1 Register**

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

Bit 7 Unimplemented, read as "0"

Bit 6~4 **PCS16~PCS14**: PC3 pin-shared function selection
000: PC3/TC1
001: AN11
010: C3VOD
011: C4VOD
100: Reserved
101~111: PC3/TC1

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PCS12~PCS10**: PC2 pin-shared function selection
000: PC2
001: LVDIN
010: C4VO
011: C2VOD
100: SDA
101: Reserved
110: CP5N
111: PC2

• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	D6	D5	IFS4	IFS3	IFS2	IFS1	IFS0
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 6~5 **D6~D5**: Reserved, must be fixed at "0"

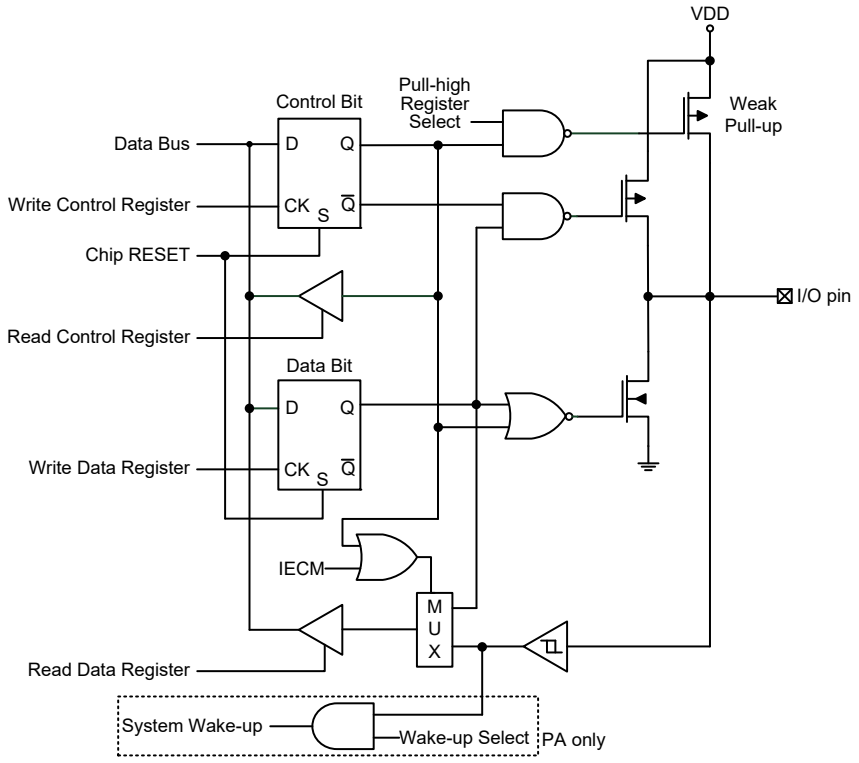
Bit 4 **IFS4**: TC1 input source pin selection
0: PB1
1: PC3

Bit 3~2 **IFS3~IFS2**: SDA input source pin selection
00: PA0
01: PB1
10: PB0
11: PC2

Bit 1~0 **IFS1~IFS0**: SCL input source pin selection
00: PA2
01: PA0
10: PB4
11: PC1

I/O Pin Structures

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



Logic Function Input/Output Structure

READ PORT Function

The READ PORT function is used to 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 a, 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

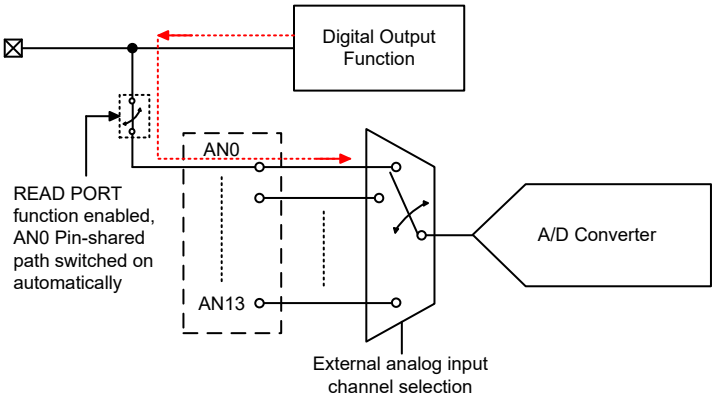
Bit 7~0 **IECS7~IECS0**: READ PORT function enable control bit 7~ bit 0
11001010: IECM=1 – READ PORT function is enabled
Others: IECM=0 – READ PORT function is disabled

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 I²C)	0			
I²C: SDA, SCL	Pin value			
Analog Function	0			

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

The additional function of the READ PORT mode is to check the A/D path. When the READ PORT function is disabled, the A/D path from the external pin to the internal analog input will be switched off if the A/D input pin function is not selected by the corresponding selection bits. For the MCU with A/D converter channels, such as A/D AN0~AN13, the desired A/D channel can be switched on by properly configuring the external analog input channel selection bits in the A/D Control Register together with the corresponding analog input pin function is selected. However, the additional function of the READ PORT mode is to force the A/D path to be switched on. For example, when the AN0 is selected as the analog input channel as the READ PORT function is enabled, the AN0 analog input path will be switched on even if the AN0 analog input pin function is not selected. In this way, the AN0 analog input path can be examined by internally connecting the digital output on this shared pin with the AN0 analog input pin switch and then converting the corresponding digital data without any external analog input voltage connected.

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



A/D Channel Input Path Internally Connection

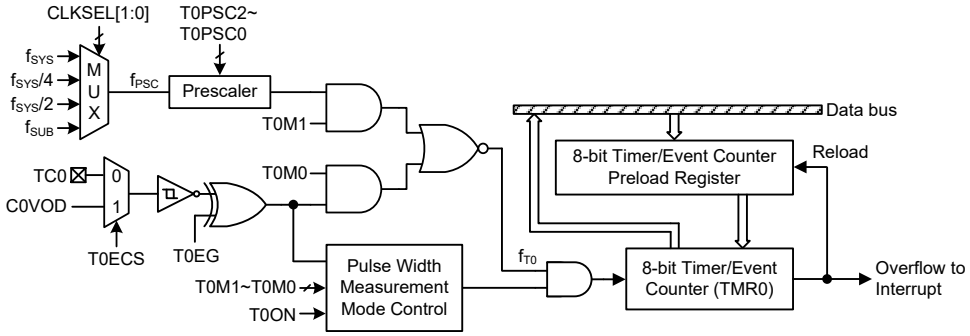
Programming Considerations

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

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

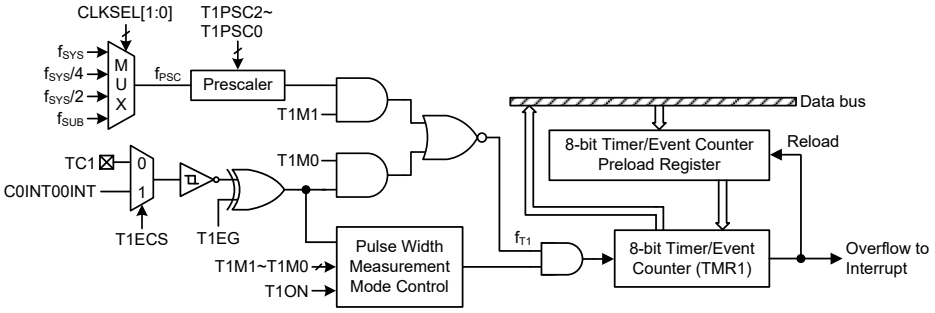
Timer/Event Counters

The provision of Timer/Event Counters forms an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contains four count-up Timer/Event Counters of 8-bit capacity. The Timer/Event Counter 0/1 have three different operating modes, they can be configured to operate as a general timer, an external event counter or a pulse width measurement device. The Timer/Event Counter 2 has two different operating modes, it can be configured to operate as a general timer or operate in the PPG non-retrigger function mode. The Timer/Event Counter 3 has four different operating modes, it can be configured to operate as a general timer, an external event counter, a pulse width measurement device or operate in the PPG retrigger function mode. The provision of an internal prescaler to the clock circuitry also gives added range to the Timer/Event Counters.



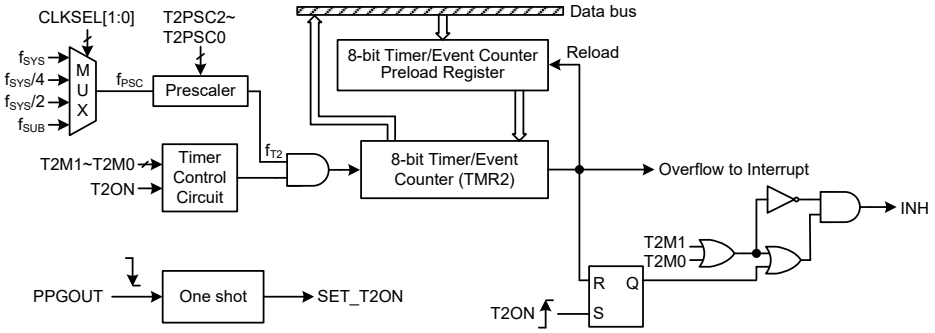
Note: The C0VOD is sourced from the Comparator 0 output (after debounce).

8-bit Timer/Event Counter 0



Note: The C0INT00INT is a high pulse signal from the Comparator 0 output. It should be noted that this signal cannot be used in pulse width measurement mode, and when used in event counter mode, the T1EG bit has no effect.

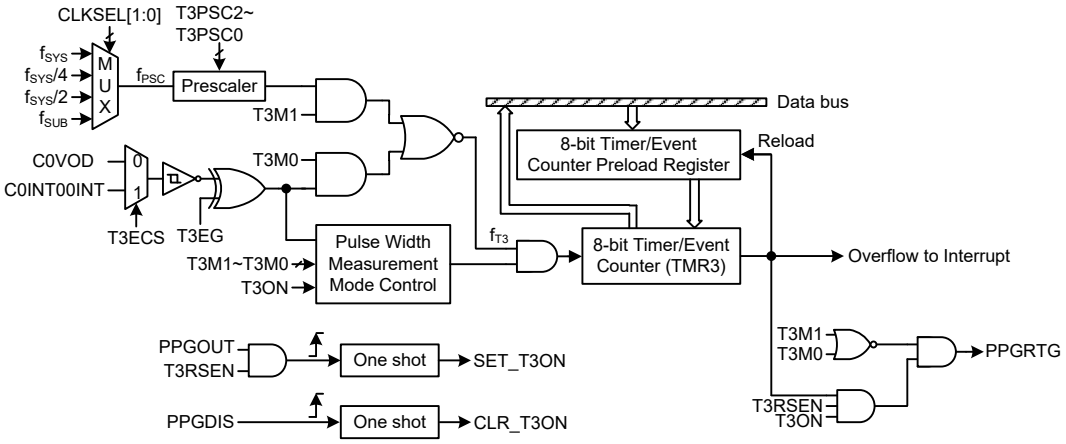
8-bit Timer/Event Counter 1



Note: 1. The PPGOUT is is sourced from the Programmable Pulse Generator output.

2. The INH is internally connected to the Programmable Pulse Generator, which can be used to inhibit further PPG triggers.

8-bit Timer/Event Counter 2



Note: 1. The C0VOD is sourced from the Comparator 0 output (after debounce).
2. The C0INT00INT is a high pulse signal from the Comparator 0 output. It should be noted that this signal cannot be used in pulse width measurement mode, and when used in event counter mode, the T3EG bit has no effect.
3. The PPGOUT and PPGDIS are sourced from the Programmable Pulse Generator outputs.
4. The PPGRTG is internally connected to the Programmable Pulse Generator, which can be used for further PPG triggers.

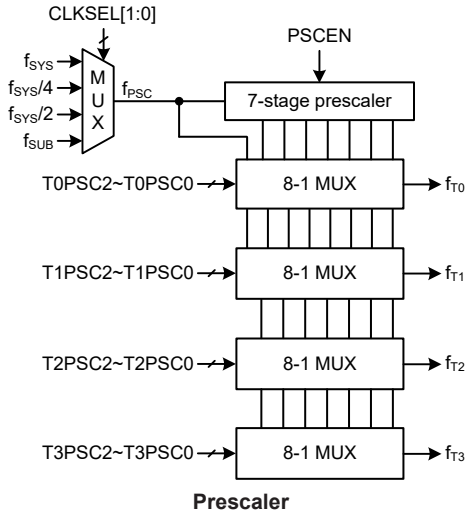
8-bit Timer/Event Counter 3

Configuring the Timer/Event Counter Input Clock Source

For the Timer/Event Counter 0/1/3, the clock source can originate from either an external clock source or an internal clock source, while for the Timer/Event Counter 2, the clock source can only be from an internal clock source. The external clock input allows the user to count external events, measure time internals or pulse widths. While using the internal clock allows the user to generate an accurate time base.

The internal clock source is provided by the internal clock f_{PSC}, which originates from the internal clock source f_{SYS}, $f_{SYS}/4$, $f_{SYS}/2$ or f_{SUB}, selected using the CLKSEL[1:0] bits in the PSCR register, and then passes through a divider. The division ratio is selected by programming the TnPSC2~TnPSC0 bits in the TMRnC register or the T3PSC2~T3PSC0 bits in the TMR3C0 register.

The external clock source can be supplied on the TC0 pin or the C0VOD signal, the TC1 pin or the C0INT00INT signal, and the C0VOD or C0INT00INT signal depending upon which Timer/Event Counter is used, the choice of which is determined by the T0ECS/T1ECS/T3ECS bit in the TMR0C/TMR1C/TMR3C0 register.



• **PSCR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PSCEN	CLKSEL1	CLKSEL0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **PSCEN**: Prescaler clock enable control
0: Disable
1: Enable
The PSCEN bit is the Prescaler clock enable or disable control bit. When the Prescaler clock is disabled, it can reduce extra power consumption.

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source selection
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/2$
11: f_{SUB}

Timer/Event Counter Register Description

There are two types of registers related to the Timer/Event Counters. The first are the registers that contain the actual value of the timer and into which an initial value can be preloaded. Reading from these registers retrieves the contents of the Timer/Event Counters. The second type of associated registers is the Timer/Event Counter control registers which define the Timer/Event Counter options and determines how the Timer/Event Counters are to be used.

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMR0C	T0M1	T0M0	T0ECS	T0ON	T0EG	T0PSC2	T0PSC1	T0PSC0
TMR0	D7	D6	D5	D4	D3	D2	D1	D0
TMR1C	T1M1	T1M0	T1ECS	T1ON	T1EG	T1PSC2	T1PSC1	T1PSC0
TMR1	D7	D6	D5	D4	D3	D2	D1	D0
TMR2C	T2M1	T2M0	—	T2ON	—	T2PSC2	T2PSC1	T2PSC0
TMR2	D7	D6	D5	D4	D3	D2	D1	D0
TMR3C0	T3M1	T3M0	T3ECS	T3ON	T3EG	T3PSC2	T3PSC1	T3PSC0

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMR3C1	D7	D6	—	—	—	—	—	T3RSEN
TMR3	D7	D6	D5	D4	D3	D2	D1	D0

Timer/Event Counter Register List

Timer/Event Counter Registers – TMR0, TMR1, TMR2, TMR3

The Timer/Event Counter registers, TMRn, are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH at which point the timer overflows and an internal interrupt signal is generated. Then the timer value will be reset with the initial preload register value and continue counting.

Note that to achieve a maximum full range count of FFH, all the preload registers must first be cleared to zero. It should be noted that after power-on, the preload registers will be in an unknown condition. If the Timer/Event Counter is in an OFF condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter the next time an overflow occurs.

- **TMRn Register (n=0~3)**

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

Bit 7~0 **D7~D0**: Timer/Event Counter n preload register bit 7 ~ bit 0

Timer/Event Counter Control Registers – TMR0C, TMR1C, TMR2C, TMR3C0/TMR3C1

The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in several different modes, the options of which are determined by the contents of their respective control registers.

For the Timer/Event Counter 0~2, the Timer Control Registers are known as TMRnC. For the Timer/Event Counter 3, the Timer Control Registers are a pair of registers and known as TMR3C0/TMR3C1. The Timer Control Registers together with their corresponding timer registers control the full operation of the Timer/Event Counters. Before the timers can be used, it is essential that the Timer Control Registers are fully programmed with the right data to ensure their correct operation, a process that is normally carried out during program initialisation.

To select which of the several modes the Timer/Event Counters are to operate in, either in the Timer Mode, the Event Counter Mode, the Pulse Width Measurement Mode, the PPG Non-retrigger Function Mode or the PPG Retrigger Function Mode, the TnM1/TnM0 bits in the TMRnC register or the T3M1/T3M0 bits in the TMR3C0 register must be set to the required logic levels. The timer-on bit, which is known as TnON, provides the basic on/off control of the respective Timer/Event Counter. Setting the bit high allows the counter to run. Clearing the bit stops the counter. The TnPSC2~TnPSC0 bits determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or pulse width measurement mode, the active transition edge level type is selected by the logic level of the TnEG bit. For the Timer/Event Counter 0/1/3, the TnECS bit is used to select the external clock source.

In addition, the T3RSEN bit in the TMR3C1 register is only available for the Timer/Event Counter 3 and related to the PPG retrigger function mode configuration.

• **TMR0C Register**

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	T0ECS	T0ON	T0EG	T0PSC2	T0PSC1	T0PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7~6 **T0M1~T0M0**: Timer/Event Counter 0 operation mode selection
00: No mode available
01: Event counter mode
10: Timer mode
11: Pulse width measurement mode

Bit 5 **T0ECS**: Timer/Event Counter 0 external clock source selection
0: TC0 pin
1: C0VOD

Bit 4 **T0ON**: Timer/Event Counter 0 counting enable control
0: Disable
1: Enable

Bit 3 **T0EG**: Timer/Event Counter 0 active edge selection
In Event Counter Mode:
0: Count on rising edge
1: Count on falling edge
In Pulse Width Measurement Mode:
0: Start counting on the falling edge, stop on rising edge
1: Start counting on the rising edge, stop on falling edge

Bit 2~0 **T0PSC2~T0PSC0**: Timer/Event Counter 0 prescaler rate selection
Timer/Event Counter 0 internal clock f_{T0}=
000: f_{PSC}
001: $f_{PSC}/2$
010: $f_{PSC}/4$
011: $f_{PSC}/8$
100: $f_{PSC}/16$
101: $f_{PSC}/32$
110: $f_{PSC}/64$
111: $f_{PSC}/128$

• **TMR1C Register**

Bit	7	6	5	4	3	2	1	0
Name	T1M1	T1M0	T1ECS	T1ON	T1EG	T1PSC2	T1PSC1	T1PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7~6 **T1M1~T1M0**: Timer/Event Counter 1 operation mode selection
00: No mode available
01: Event counter mode
10: Timer mode
11: Pulse width measurement mode

Bit 5 **T1ECS**: Timer/Event Counter 1 external clock source selection
0: TC1 pin
1: C0INT00INT
Note that the C0INT00INT is a high pulse signal from comparator 0 that cannot be used in pulse width measurement mode. When this signal is used in event counter mode, the T1EG bit has no effect.

Bit 4 **T1ON**: Timer/Event Counter 1 counting enable control
0: Disable
1: Enable

Bit 3 **T1EG**: Timer/Event Counter 1 active edge selection
In Event Counter Mode:
0: Count on rising edge
1: Count on falling edge
In Pulse Width Measurement Mode:
0: Start counting on the falling edge, stop on rising edge
1: Start counting on the rising edge, stop on falling edge

Bit 2~0 **T1PSC2~T1PSC0**: Timer/Event Counter 1 prescaler rate selection
Timer/Event Counter 0 internal clock f_{T1}=
000: f_{PSC}
001: $f_{PSC}/2$
010: $f_{PSC}/4$
011: $f_{PSC}/8$
100: $f_{PSC}/16$
101: $f_{PSC}/32$
110: $f_{PSC}/64$
111: $f_{PSC}/128$

- **TMR2C Register**

Bit	7	6	5	4	3	2	1	0
Name	T2M1	T2M0	—	T2ON	—	T2PSC2	T2PSC1	T2PSC0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

Bit 7~6 **T2M1~T2M0**: Timer/Event Counter 2 operation mode selection
00: PPG non-retrigger function mode
01: No mode available
10: Timer mode
11: No mode available

Bit 5 Unimplemented, read as "0"

Bit 4 **T2ON**: Timer/Event Counter 2 counting enable control
0: Disable
1: Enable
This bit cannot be modified by the application program in the PPG non-retrigger function mode to avoid the PPG abnormal operations.

Bit 3 Unimplemented, read as "0"

Bit 2~0 **T2PSC2~T2PSC0**: Timer/Event Counter 2 prescaler rate selection
Timer/Event Counter 2 internal clock f_{T2}=
000: f_{PSC}
001: $f_{PSC}/2$
010: $f_{PSC}/4$
011: $f_{PSC}/8$
100: $f_{PSC}/16$
101: $f_{PSC}/32$
110: $f_{PSC}/64$
111: $f_{PSC}/128$

• **TMR3C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	T3M1	T3M0	T3ECS	T3ON	T3EG	T3PSC2	T3PSC1	T3PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7~6 **T3M1~T3M0**: Timer/Event Counter 3 operation mode selection
00: PPG retrigger function mode
01: Event counter mode
10: Timer mode
11: Pulse width measurement mode

Bit 5 **T3ECS**: Timer/Event Counter 3 external clock source selection
0: C0VOD
1: C0INT00INT
The C0INT00INT is a high pulse signal from comparator 0 that cannot be used in pulse width measurement mode. When this signal is used in event counter mode, the T3EG bit has no effect.

Bit 4 **T3ON**: Timer/Event Counter 3 counting enable control
0: Disable
1: Enable

Bit 3 **T3EG**: Timer/Event Counter 3 active edge selection
In Event Counter Mode:
0: Count on rising edge
1: Count on falling edge
In Pulse Width Measurement Mode:
0: Start counting on the falling edge, stop on rising edge
1: Start counting on the rising edge, stop on falling edge

Bit 2~0 **T3PSC2~T3PSC0**: Timer/Event Counter 3 prescaler rate selection
Timer/Event Counter 3 internal clock f_{T3}=
000: f_{PSC}
001: $f_{PSC}/2$
010: $f_{PSC}/4$
011: $f_{PSC}/8$
100: $f_{PSC}/16$
101: $f_{PSC}/32$
110: $f_{PSC}/64$
111: $f_{PSC}/128$

• **TMR3C1 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	—	—	—	—	—	T3RSEN
R/W	R/W	R/W	—	—	—	—	—	R/W
POR	0	0	—	—	—	—	—	0

Bit 7~6 Reserved, this bit must be fixed at “0”

Bit 5~1 Unimplemented, read as “0”

Bit 0 **T3RSEN**: Restart the PPG counter using Timer/Event Counter 3 overflow (PPG retrigger function mode) enable control
0: Disable
1: Enable
When restarting the PPG counter using Timer/Event Counter 3 overflow is disabled, the PPG module output can be restarted by the software control bit, PST, or other hardware trigger only. When restarting the PPG counter using Timer/Event Counter 3 overflow is enabled, the PPG module can be restarted by Timer/Event Counter 3 overflow, other hardware trigger or software control by setting the PST to 1.

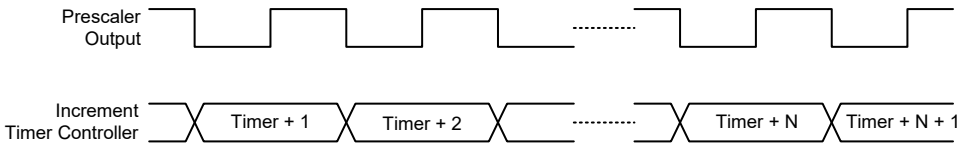
Timer/Event Counter Operating Modes

The Timer/Event Counters can operate in different operating modes, any in the timer mode, the event counter mode, the pulse width measurement mode, the PPG non-retrigger function mode or the PPG retrigger function mode. The operating mode is selected using the TnM1~TnM0 bits in the TMRnC register or the T3M1~T3M0 bits in the TMR3C0 register.

Timer Mode

In this mode, the Timer/Event Counter n can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter n overflows. To operate in this mode, the TnM1~TnM0 bits in the TMRnC register or the T3M1~T3M0 bits in the TMR3C0 register must be set to 10 respectively.

In this mode the internal clock is used as the timer clock. The Timer/Event Counter n clock is clock source, f_{Tn}, which is sourced from the internal clock f_{PSC}. The f_{PSC} originates from the internal clock source f_{SYS}, $f_{SYS}/4$, $f_{SYS}/2$ or f_{SUB}, which is selected using the CLKSEL[1:0] bits in the PSCR register, and then passes through a divider, the division ratio of which is selected by programming the TnPSC2~TnPSC0 bits in the TMRnC register or the T3PSC2~T3PSC0 bits in the TMR3C0 register. The timer-on bit, TnON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one. When the timer is full and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupts are two of the wake-up sources. However, the internal interrupts can be disabled by ensuring that the Timer/Event Counter n interrupt enable bits in the corresponding interrupt control registers are reset to zero.



Timer Mode Timing Chart

Event Counter Mode

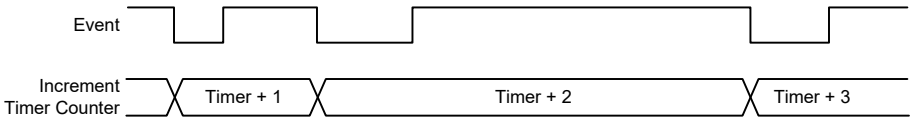
This mode only exists in the Timer/Event Counter 0/1/3. In this mode, a number of changing logic events, occurring on the TC0 pin or C0VOD signal, the TC1 pin or C0INT00INT signal, and the C0VOD or C0INT00INT signal, can be recorded by the Timer/Event Counter 0/1/3 respectively. To operate in this mode, the T0M1~T0M0 bits in the TMR0C register, the T1M1~T1M0 bits in the TMR1C register or the T3M1~T3M0 bits in the TMR3C0 register must be set to 01 respectively.

In this mode, the TC0 pin or C0VOD signal, the TC1 pin or C0INT00INT signal, and the C0VOD or C0INT00INT signal can be used as the Timer/Event Counter 0/1/3 clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit, T0ON/T1ON/T3ON, in the TMR0C/TMR1C/TMR3C0 register, can be set high to enable the Timer/Event Counter 0/1/3 to run. If the active edge selection bit, T0EG/T1EG/T3EG, in the TMR0C/TMR1C/TMR3C0 register, is low, the Timer/Event Counter 0/1/3 will increment each time the TC0 pin or C0VOD signal, the TC1 pin or C0INT00INT signal, and the C0VOD or C0INT00INT signal receives a low to high transition. If the T0EG/T1EG/T3EG is high, the counter will increment each time the TC0 pin or C0VOD signal, the TC1 pin or C0INT00INT signal, and the C0VOD or C0INT00INT signal receives a high to low transition. Note that for the Timer/Event Counter 1/3, when the C0INT00INT signal is used in event counter mode, the T1EG/T3EG bit has no effect. When the Timer/Event Counter 0/1/3 is full and overflows, an interrupt signal is generated and the Timer/Event Counter 0/1/3 will reload the value already loaded into the preload register and

continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter 0/1/3 interrupt enable bit in the corresponding interrupt control register is reset to zero.

It should be noted that in the event counter mode, even if the microcontroller is in the SLEEP or IDLE Mode, the Timer/Event Counter 0/1/3 will continue to record externally changing logic events on the TC0 pin or C0VOD signal, the TC1 pin or C0INT00INT signal, and the C0VOD or C0INT00INT signal. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.

As the TC0 or TC1 pin is shared with an I/O pin, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the Operating Mode Selection bits in the Timer Control Register place the Timer/Event Counter 0/1 in the Event Counter Mode. The second is to ensure that the port control register configures the pin as an input.



Event Counter Mode Timing Chart – TnEG=1 (n=0/1/3)

Pulse Width Measurement Mode

This mode only exists in the Timer/Event Counter 0/1/3. In this mode, the Timer/Event Counter 0/1/3 can be utilised to measure the width of pulses applied to the TC0 pin or C0VOD signal, the TC1 pin, and the C0VOD signal. To operate in this mode, the T0M1~T0M0 bits in the TMR0C register, the T1M1~T1M0 bits in the TMR1C register or the T3M1~T3M0 bits in the TMR3C0 register must be set to 11 respectively.

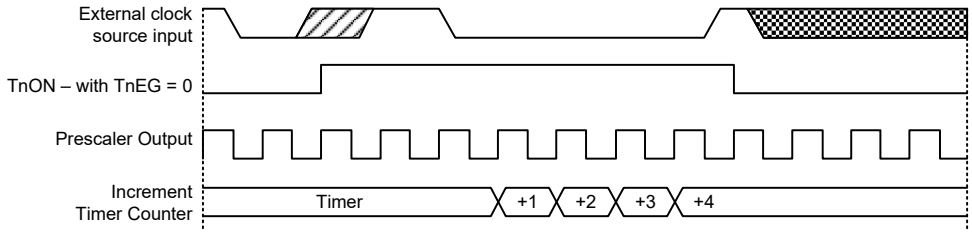
In this mode, the internal clock f_{PSC} is used as the timer clock, which can be selected to be derived from f_{SYS}, $f_{SYS}/4$, $f_{SYS}/2$ or f_{SUB} by setting the CLKSEL[1:0] bits in the PSCR register. The division of the f_{PSC} clock is selected by the T0PSC[2:0]/T1PSC[2:0]/T3PSC[2:0] bits in the TMR0C/TMR1C/TMR3C0 register. After the other bits in the Timer Control Register have been setup, the enable bit T0ON/T1ON/T3ON, in the TMR0C/TMR1C/TMR3C0 register, can be set high to enable the Timer/Event Counter 0/1/3 to run. However it will not actually start counting until an active edge is received on the TC0 pin or C0VOD signal, the TC1 pin, and the C0VOD signal.

If the active edge selection bit, T0EG/T1EG/T3EG, in the TMR0C/TMR1C/TMR3C0 register, is low, once a high to low transition has been received on the TC0 pin or C0VOD signal, the TC1 pin, and the C0VOD signal, the Timer/Event Counter 0/1/3 will start counting until the TC0 pin or C0VOD signal, the TC1 pin, and the C0VOD signal returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter 0/1/3 will stop counting. If the active edge selection bit T0EG/T1EG/T3EG is high, the Timer/Event Counter 0/1/3 will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will be automatically reset to zero and the Timer/Event Counter 0/1/3 will stop counting. It is important to note that in the pulse width measurement mode, the enable bit is automatically reset to zero when the TC0 pin or C0VOD signal, the TC1 pin, and the C0VOD signal returns to its original level, whereas in the other modes the enable bit can only be reset to zero under program control.

The residual value in the Timer/Event Counter 0/1/3, which can now be read by the program, therefore represents the length of the pulse received on the TC0 pin or C0VOD signal, the TC1 pin, and the C0VOD signal. As the enable bit has now been reset, any further transitions on the TC0 pin or C0VOD signal, the TC1 pin, and the C0VOD signal will be ignored. The timer cannot begin further pulse width measurement until the enable bit is set high again by the program. In this way, single shot pulse measurements can be easily made.

It should be noted that in this mode the Timer/Event Counter 0/1/3 is controlled by logical transitions on the TC0 pin or C0VOD signal, the TC1 pin, and the C0VOD signal and not by the logic level. When the Timer/Event Counter 0/1/3 is full and overflows, an interrupt signal is generated and the Timer/Event Counter 0/1/3 will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter 0/1/3 interrupt enable bit in the corresponding interrupt control register is reset to zero.

As the TC0 or TC1 pin is shared with an I/O pin, to ensure that the pin is configured to operate as a pulse width measurement pin, two things have to be implemented. The first is to ensure that the operating mode selection bits in the Timer Control Register place the Timer/Event Counter 0/1 in the pulse width measurement mode, the second is to ensure that the port control register configure the pin as an input.



Pulse Width Measurement Mode Timing Chart – TnEG=0 (n=0/1/3)

PPG Non-retrigger Function Mode

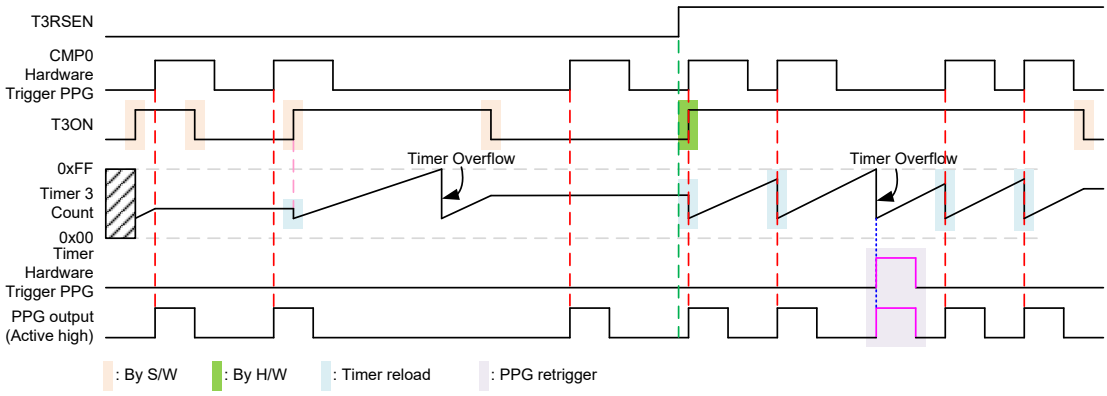
The Timer/Event Counter 2 has a PPG non-retrigger function mode for PPG usage. This mode is used to implement the PPG non-retrigger function. To operate in this mode, the T2M1~T2M0 bits in the TMR2C register must be set to 00 respectively.

In this mode, the Timer/Event Counter 2 starts counting when PPG is stopped and stops when overflow. That means the T2ON will be set once the PPG is stopped and cleared when overflow. Once an overflow occurs, the counter is reloaded from the Timer/Event Counter 2 preload register, and generates an interrupt request flag. The interrupt can be disabled by ensuring that the Timer/Event Counter 2 interrupt enable bit in the corresponding interrupt control register is reset to zero.

PPG Retrigger Function Mode

The Timer/Event Counter 3 has a PPG retrigger function mode for PPG usage. This mode is used to implement the PPG retrigger function. To operate in this mode, the T3M1~T3M0 bits in the TMR3C0 register must be set to 00 respectively.

When T3M1/T3M0=00, the PPG retrigger function is related to the T3RSEN bit setting. When T3RSEN=0 and T3ON=1, the Timer/Event Counter 3 overflow will not trigger a PPGRTG signal. If T3ON=0, when the PPGOUT signal, which is an inactive to active transition, is generated, the T3ON will not be triggered and remained unchanged. When T3RSEN=1 and T3ON=1, the Timer/Event Counter 3 overflow will trigger a PPGRTG signal. If T3ON=0, when the PPGOUT signal, which is an inactive to active transition, is generated, the T3ON will be triggered from 0 to 1, and the TMR3 register will be reloaded.



PPG Retrigger Function Mode Timing Chart

T3M[1:0]	T3RSEN	T3ON	PPGOUT	PPGDIS	State
00	0	0	x	x	The Timer/Event Counter 3 has no action
		1	Inactive to active	x	Forbidden
		1	Except inactive to active	x	Forbidden
	1	0	Inactive to active	0	The T3ON is set high by hardware, the Timer/ Event Counter 3 is reloaded
		1	Inactive to active	0	The Timer/Event Counter 3 is reloaded
		1	Except inactive to active	0	When the Timer/Event Counter 3 overflows, the PPG is retriggered and the Timer/Event Counter 3 is reloaded
		1	x	1	The T3ON is cleared to zero by hardware

"x": Don't care

I/O Interfacing

The Timer/Event Counter 0/1, when configured to run in the event counter or pulse width measurement mode, can use an external pin for its operation. The external pin TC0/TC1 is used as the Timer/Event Counter 0/1 clock source by clearing the T0ECS/T1ECS bit to zero. As the TC0/TC1 pin is a shared pin it must be configured correctly to ensure that it is setup for use as a Timer/Event Counter input pin. Additionally the corresponding Port Control Register bit must be set high to ensure that the pin is setup as an input. Any pull-high resistor connected to this pin will remain valid even if the pin is used as a Timer/Event Counter input.

Programming Considerations

When the Timer/Event Counter is configured to run in the timer mode, the internal system clock is used as the timer clock source and is therefore synchronised with the overall operation of the microcontroller. In this mode when the appropriate timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector. For the pulse width measurement mode, the internal system clock is also used as the timer clock source but the Timer/Event Counter will only run when the correct logic condition appears on the TC0 pin or C0VOD signal, the TC1 pin, and the C0VOD signal. As this is an event and not synchronised with the internal timer clock, the microcontroller will only see this event when the next timer clock pulse arrives. As a result, there may be small differences in measured values requiring programmers to take this into account during programming. The same applies if the Timer/Event Counter is configured to be in the event counter mode, which again is an event and not synchronised with the internal system or timer clock.

When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, this should be taken into account by the programmer. Care must be taken to ensure that the Timer/Event Counters are properly initialised before using them for the first time. The associated timer interrupt enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the Timer/Event Counter will remain inactive. The active edge, operating mode and clock source and prescaler rate selection bits in timer control register must also be correctly set to ensure the Timer/Event Counter is properly configured for the required application. It is also important to ensure that an initial value is first loaded into the timer registers before the Timer/Event Counter is switched on; this is because after power-on the initial values of the timer registers are unknown. After the Timer/Event Counter has been initialized the Timer/Event Counter can be turned on and off by controlling the enable bit in the timer control register.

When the Timer/Event Counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the Timer/Event Counter interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in the SLEEP or IDLE mode. This situation may occur if the Timer/Event Counter is in the Event Counter Mode and if the TC0 pin or C0VOD signal, the TC1 pin or C0INT00INT signal, and the C0VOD or C0INT00INT signal continues to change state. In such a case, the Timer/Event Counter will continue to count these external events and if an overflow occurs the device will be woken up from its Power-down condition. To prevent such a wake-up from occurring, the Timer/Event Counter interrupt request flag should first be set high before issuing the “HALT” instruction to enter the SLEEP or IDLE Mode.

Timer/Event Counter Program Example

This program example shows how the Timer/Event Counter registers are setup, along with how the interrupts are enabled and managed. Note how the Timer/Event Counter is turned on, by setting the TnON bit in the Timer Control Register. The Timer/Event Counter can be turned off in a similar way by clearing the same bit. This example program sets the Timer/Event Counters 0 to be in the timer mode, which uses the internal system clock as its clock source.

Timer/Event Counter Programming Example

```
org 0ch              ; CMP0 INT00 interrupt vector
org 10h              ; Timer/Event Counter 0 interrupt vector
jmp tmr0int          ; jump here when Timer/Event Counter 0 overflows
:
org 20h              ; main program
:
                     ; internal Timer/Event Counter 0 interrupt routine
tmr0int:
:
                     ; Timer/Event Counter 0 main program placed here
begin:
                     ; setup Timer/Event Counter 0 registers
mov a, 09bh          ; setup Timer/Event Counter 0 preload value
mov tmr0, a
mov a, 081h          ; setup Timer/Event Counter 0 control register
mov tmr0c, a         ; timer mode and prescaler set to /2
                     ; setup interrupt register
mov a, 001h          ; enable both master and Timer/Event Counter 0 interrupts
mov intc0, a
mov a, 001h
mov intc1, a
:
set tmr0c.4          ; start Timer/Event Counter 0
```

Analog to Digital Converter

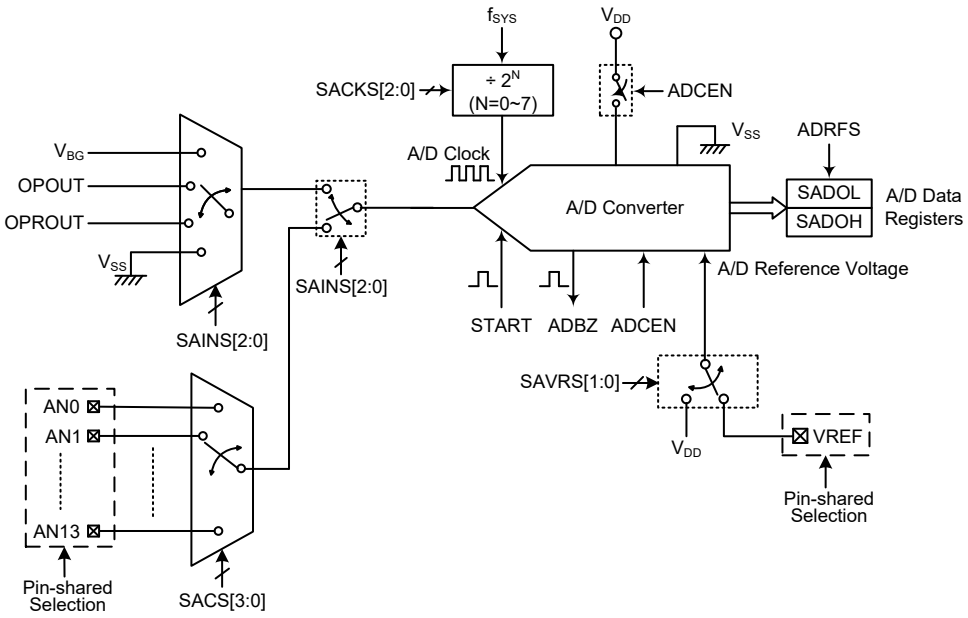
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

A/D Converter Overview

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the internal Bandgap reference voltage, V_{BG}, the operational amplifier output, OPOUT, the operational amplifier output, OPROUT, or the A/D converter negative power supply, V_{SS}, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 and SACS3~SACS0 bits. Note that when the internal analog signal is selected to be converted, the external channel analog input will be automatically switched off. More detailed information about the A/D converter input signal is described in the "A/D Converter Control Registers" and "A/D Converter Input Signals" sections respectively.

External Input Channels	Internal Signals
14: AN0~AN13	V_{BG}, OPOUT, OPROUT, V_{SS}

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



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using a series of registers. A read only register pair exists to store the A/D converter data 12-bit value. The remain two control registers which setup the operating conditions and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

A/D Converter Register List

A/D Converter Data Registers – SADOL, SADOH

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

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

A/D Converter Data Registers

A/D Converter Control Registers – SADC0, SADC1

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

The relevant pin-shared function selection bits determine which pins on I/O Ports are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

• **SADC0 Register**

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

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

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

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

Bit 4 **ADRFS**: A/D Converter data format control
0: ADC output data format → SADOH=D[11:4]; SADOL=D[3:0]
1: ADC output data format → SADOH=D[11:8]; SADOL=D[7:0]
This bit controls the format of the 12-bit converted A/D converter value in the two A/D converter data registers.

Bit 3~0 **SACS3~SACS0**: A/D converter external analog channel input select
0000: AN0
0001: AN1
:
1011: AN11
1100: AN12
1101: AN13
1110~1111: Undefined, input floating

• **SADC1 Register**

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

Bit 7~5 **SAINS2~SAINS0**: A/D converter input signal select
000: External signal – External analog channel input, ANn
001: Internal signal – V_{BG} reference voltage, V_{BG}
010: Internal signal – Operational amplifier output, OPOUT
011: Internal signal –Operational amplifier output, OPROUT
100: Internal input – A/D converter negative power supply, V_{SS}
101~111: External signal – External analog channel input, ANn
When the internal analog signal is selected to be converted, the external channel input signal will automatically be switched off regardless of the SACS3~SACS0 bit value. It will prevent the external channel input from being connected together with the internal analog signal.

Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage selection
00: External VREF pin input
01: Internal A/D converter power, V_{DD}
1x: External VREF pin input
These bits are used to select the A/D converter reference voltage. When the internal A/D converter power is selected as the reference voltage, the hardware will automatically disconnect the external VREF input.

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source (f_{ADCK}) selection
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

A/D Converter Reference Voltage

The actual reference voltage supply to the A/D Converter can be supplied from the positive power supply, V_{DD} or an external reference source supplied on pin VREF. The desired selection is made using the SAVRS1~SAVRS0 bits in the SADC1 register. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage input pin, the VREF pin-shared function selection bits should first be properly configured to disable other pin-shared functions. However, if the internal reference signal is selected as the reference source, the external reference input from the VREF pin will automatically be switched off by hardware.

Note that the analog input signal values must not be allowed to exceed the value of the selected A/D Converter reference voltage.

SAVRS[1:0]	Reference Source	Description
00, 1x	VREF pin	External A/D converter reference pin VREF
01	V_{DD}	Internal A/D converter power supply voltage

A/D Converter Reference Voltage Selection

A/D Converter Input Signals

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

The A/D Converter also has four internal anlog signals, the Bandgap reference voltage, V_{BG}, the operational amplifier output, OPOUT, the operational amplifier output, OPROUT, or the A/D converter negative power supply, V_{SS}. As this device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter.

The SAINS2~SAINS0 bits in the SADC1 register are used to determine whether the analog signal to be converted comes from an external channel input or internal analog signal. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. If the SAINS2~SAINS0 bits are set to "000", "101"~"111" the external channel input will be selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected.

If the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off. It will prevent the external channel input from being connected together with the internal analog signal.

SAINS[2:0]	SACS[3:0]	Input Signals	Description
000, 101~111	0000~1101	AN0~AN13	External pin analog input
	1110~1111	—	Non-existed channel, input is floating
001	xxxx	V_{BG}	Internal Bandgap reference voltage
010	xxxx	OPOUT	Operational amplifier output
011	xxxx	OPROUT	Operational amplifier output
100	xxxx	V_{SS}	A/D converter negative power supply

A/D Converter Input Signal Selection

A/D Converter Clock Source

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the SACKS2~SACKS0 bits in the SADC1 register. Although the A/D conversion clock source is determined by the system clock f_{SYS}, and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D conversion clock source speed that can be selected. As the recommended value of permissible A/D conversion clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken when selecting the clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the SACKS2~SACKS0 bits should not be set to "000", "110" or "111". Doing so will give A/D conversion clock periods that are less than the minimum A/D conversion clock period or greater than the maximum A/D conversion clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * special care must be taken.

f_{SYS}	A/D Conversion Clock Period (t_{ADCK})							
	SACKS[2:0] = 000 (f_{SYS})	SACKS[2:0] = 001 ($f_{SYS}/2$)	SACKS[2:0] = 010 ($f_{SYS}/4$)	SACKS[2:0] = 011 ($f_{SYS}/8$)	SACKS[2:0] = 100 ($f_{SYS}/16$)	SACKS[2:0] = 101 ($f_{SYS}/32$)	SACKS[2:0] = 110 ($f_{SYS}/64$)	SACKS[2:0] = 111 ($f_{SYS}/128$)
1MHz	1μs	2μs	4μs	8μs	16μs*	32μs*	64μs*	128μs*
2MHz	500ns	1μs	2μs	4μs	8μs	16μs*	32μs*	64μs*
4MHz	250ns*	500ns	1μs	2μs	4μs	8μs	16μs*	32μs*
8MHz	125ns*	250ns*	500ns	1μs	2μs	4μs	8μs	16μs*
12MHz	83ns*	167ns*	333ns*	667ns	1.33μs	2.67μs	5.33μs	10.67μs*
16MHz	62.5ns*	125ns*	250ns*	500ns	1μs	2μs	4μs	8μs

A/D Conversion Clock Period Examples

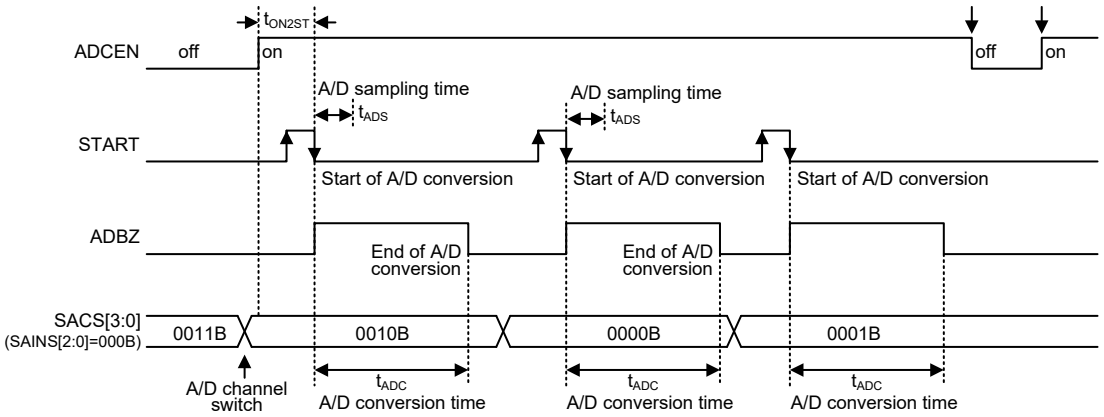
Controlling the power on/off function of the A/D conversion circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D conversion internal circuitry a certain delay, as indicated in the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

A/D Conversion Rate and Timing Diagram

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

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

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



A/D Conversion Timing

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by properly programming the SACKS2~SACKS0 bits in the SADC1 register.
- Step 2
  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to "1".
- Step 3
  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS2~SAINS0 bit in the SADC1 register.
  Select the external channel input to be converted, go to Step 4.
  Select the internal analog signal to be converted, go to Step 5.
- Step 4
  If the external channel input is selected, the desired external channel input is selected by configuring the SACS3~SACS0 bits. When the A/D input signal comes from the external channel input, the corresponding pin should be configured as an A/D input function by configuring the relevant pin-shared function control bits. Then go to Step 6.

- Step 5
  If the A/D input signal is selected to come from the internal analog signal by configuring the SAINS2~SAINS0 and the external channel analog signal input will be automatically switched off regardless of the SACS3~SACS0 bits value. After this step, go to Step 6.
- Step 6
  Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register.
- Step 7
  Select the A/D converter output data format by configuring the ADRFS bit in the SADC0 register.
- Step 8
  If the A/D converter interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, the A/D converter interrupt control bit, ADE, and its corresponding multi-function interrupt control bit must all be set high.
- Step 9
  The A/D conversion procedure can now be initialised by setting the START bit from low to high and then low again.
- Step 10
  If an A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process has completed, the ADBZ flag will go low after which the output data can be read from the SADOH and SADOL registers.

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

Programming Considerations

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

A/D Conversion Function

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

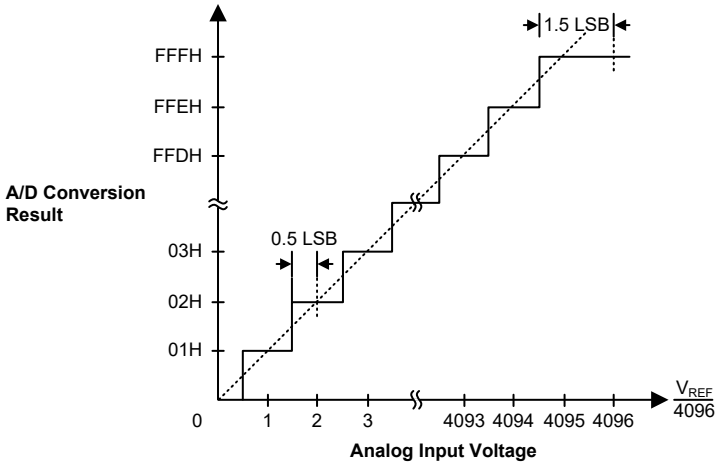
$$1\ \text{LSB}=V_{REF}\div 4096$$

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

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

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

Note that here the V_{REF} voltage is the actual A/D converter reference voltage determined by the SAVRS bit field.



Ideal A/D Conversion Function

A/D Converter Programming Examples

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

Example 1: using an ADBZ polling method to detect the end of conversion

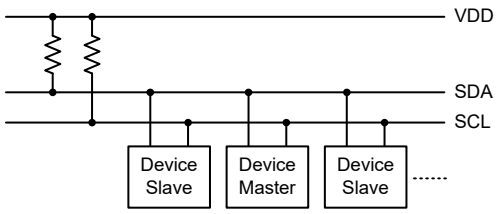
```
clr ADE                ; disable ADC interrupt
mov a,0Bh              ; select fSYS/8 as A/D clock and A/D input signal
                       ; comes from external channel
mov SADC1,a            ; select VDD as A/D reference voltage source
mov a,02h              ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,20h              ; enable A/D converter and select AN0 external channel input
mov SADC0,a
:
start_conversion:
clr START              ; high pulse on start bit to initiate conversion
set START              ; reset A/D converter
clr START              ; start A/D conversion
polling_EOC:
sz  ADBZ               ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC        ; continue polling
mov a,SADOL            ; read low byte conversion result value
mov SADOL_buffer,a     ; save result to user defined register
mov a,SADOH            ; read high byte conversion result value
mov SADOH_buffer,a     ; save result to user defined register
:
jmp start_conversion   ; start next A/D conversion
```

Example 2: using the interrupt method to detect the end of conversion

```
clr ADE               ; disable ADC interrupt
mov a,0Bh             ; select fSYS/8 as A/D clock and A/D input signal
                      ; comes from external channel
mov SADC1,a           ; select VDD as A/D reference voltage source
mov a,02h             ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,20h             ; enable A/D converter and select AN0 external channel input
mov SADC0,a
:
Start_conversion:
clr START             ; high pulse on START bit to initiate conversion
set START             ; reset A/D converter
clr START             ; start A/D conversion
clr ADF               ; clear ADC interrupt request flag
set ADE               ; enable ADC interrupt
set EMI               ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov acc_stack,a       ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a    ; save STATUS to user defined memory
:
:
mov a,SADOL           ; read low byte conversion result value
mov SADOL_buffer,a    ; save result to user defined register
mov a,SADOH           ; read high byte conversion result value
mov SADOH_buffer,a    ; save result to user defined register
:
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a          ; restore STATUS from user defined memory
mov a,acc_stack       ; restore ACC from user defined memory
```

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.

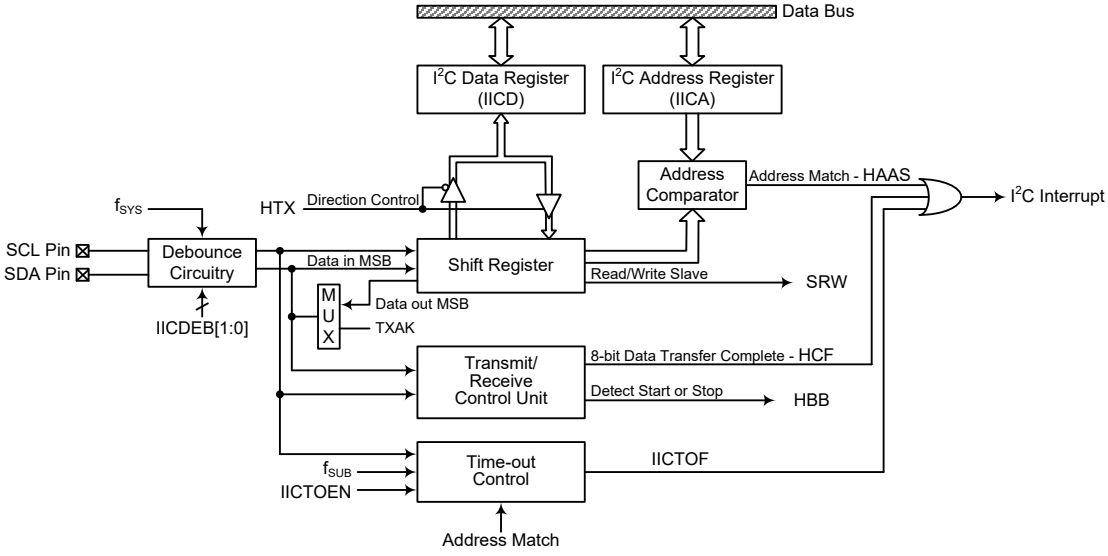


I²C Master Slave Bus Connection

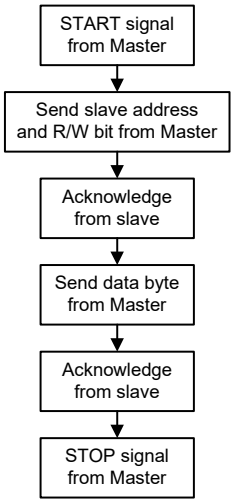
I²C Interface Operation

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 these devices, which only operate 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 the I²C device is activated and the related internal pull-high function could be controlled by its corresponding pull-high control register. It is suggested that the device should not enter the IDLE/SLEEP mode during the I²C communication.



I²C Block Diagram



I²C Interface Operation

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

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

I²C Minimum f_{SYS} Frequency Requirements

I²C Registers

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

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

I²C Register List

I²C Data Register

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

- **IICD Register**

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

"x": Unknown

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

I²C Address Register

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

- **IICA Register**

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

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

Bit 0 Unimplemented, read as "0"

I²C Control Registers

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

- **IICC0 Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **IICDEB1~IICDEB0**: I²C debounce time selection
00: Reserved
01: 2 system clock debounce
1x: 4 system clock debounce

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

Bit 1 **IICEN**: I²C enable control
0: Disable
1: Enable
The bit is the overall on/off control for the I²C interface. When the IICEN bit is cleared to zero to disable the I²C interface, the SDA and SCL lines will lose their I²C function and the I²C operating current will be reduced to a minimum value. When the bit is high the I²C interface is enabled. If the IICEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 Unimplemented, read as "0"

• IICC1 Register

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

Bit 7 **HCF**: I²C bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer
The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Bit 6 **HAAS**: I²C bus address match flag
0: Not address match
1: Address match
The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB**: I²C bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy
The HBB flag is the I²C busy flag. This flag will be "1" when the I²C bus is busy which will occur when a START signal is detected. The flag will be cleared to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX**: I²C slave device is transmitter or receiver selection
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK**: I²C bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag
The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8 bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set TXAK bit to "0" before further data is received.

Bit 2 **SRW**: I²C slave read/write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode
The SRW flag is the I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C bus. When the transmitted address and slave address 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.

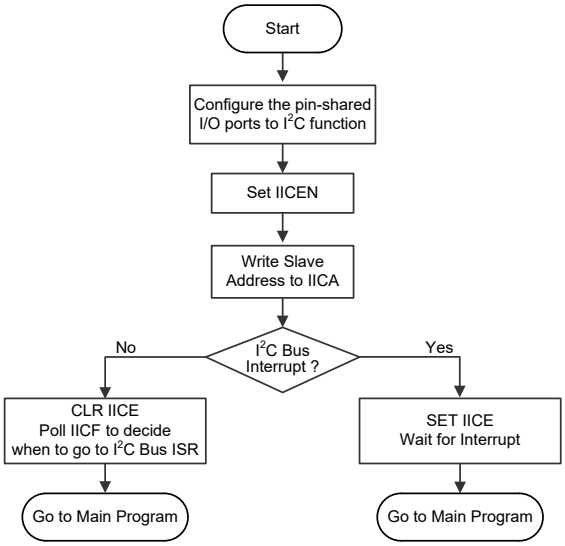
Bit 1 **IAMWU**: I²C address match wake-up control
0: Disable
1: Enable
This bit should be set to 1 to enable the I²C address match wake-up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake-up, then this bit must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

Bit 0 **RXAK**: I²C bus receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave does not receive acknowledge flag
The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

I²C Bus Communication

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

- Step 1
  Configure the corresponding pin-shared function as the I²C functional pins and set the IICEN bit in the IICC0 register to "1" to enable the I²C bus.
- Step 2
  Write the slave address of the device to the I²C bus address register IICA.
- Step 3
  Set the IICE interrupt enable bit of the interrupt control register to enable the I²C interrupt.



I²C Bus Initialisation Flowchart

I²C Bus Start Signal

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

I²C Slave Address

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

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

I²C Bus Read/Write Signal

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

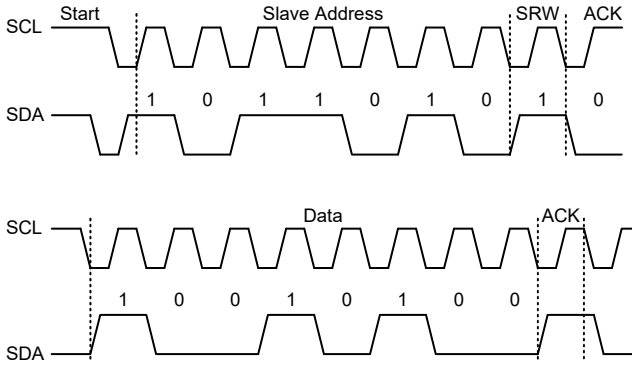
I²C Bus Slave Address Acknowledge Signal

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 set to be a transmitter so the HTX bit in the IICC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be set as a receiver and the HTX bit in the IICC1 register should be set to "0".

I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the IICD register. If set as a transmitter, the slave device must first write the data to be transmitted into the IICD register. If set as a receiver, the slave device must read the transmitted data from the IICD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The slave device, which is set as a transmitter will check the RXAK bit in the IICC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.

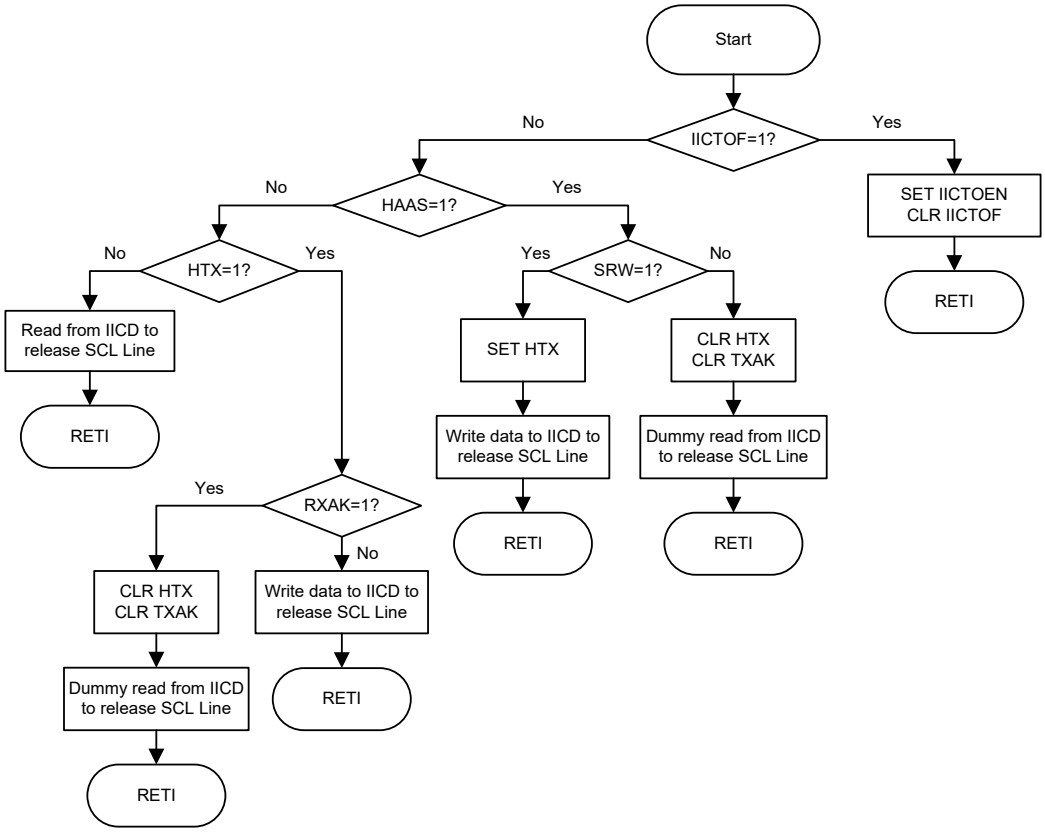


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	·····	S	SA	SR	M	D	A	D	A	·····	P

I²C Communication Timing Diagram

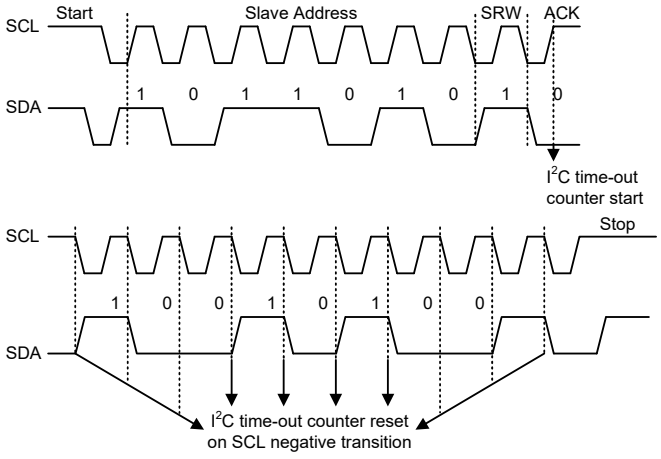
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the SCL line.



I²C Bus ISR Flowchart

I²C Time-out Control

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



I²C Time-out

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

Registers	After I²C Time-out
IICD, IICA, IICC0	No change
IICC1	Reset to POR condition

I²C Registers after Time-out

The IICTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using IICTOS5~IICTOS0 bits in the IICTOC register. The time-out time is given by the formula: $[(1\sim64)\times32]/f_{SUB}$. This gives a time-out period which ranges from about 1ms to 64ms.

- **IICTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

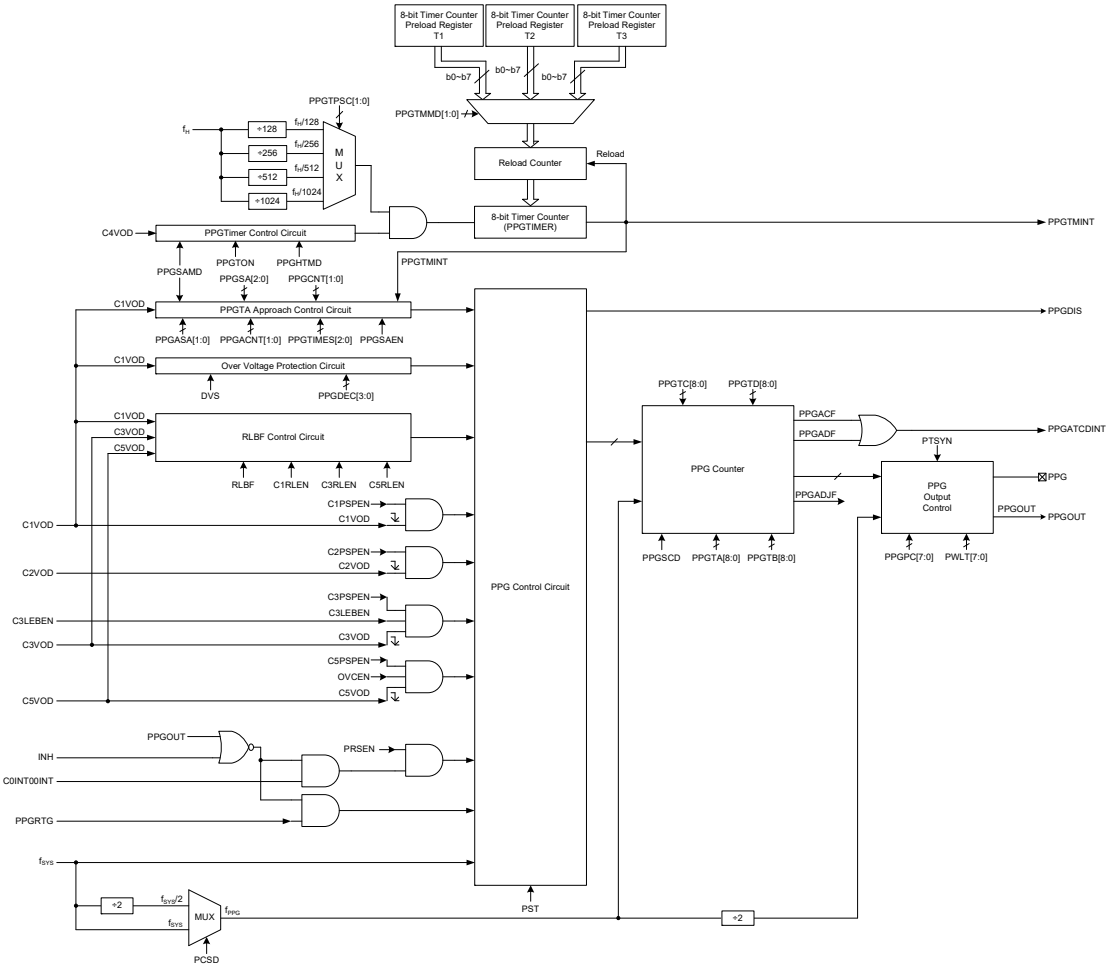
Bit 7 **IICTOEN**: I²C Time-out control
0: Disable
1: Enable

Bit 6 **IICTOF**: I²C Time-out flag
0: No time-out occurred
1: Time-out occurred
This bit is set high when time-out occurs and can only be cleared to zero by application program.

Bit 5~0 **IICTOS5~IICTOS0**: I²C Time-out period selection
I²C time-out clock source is $f_{SUB}/32$.
I²C time-out time is equal to $(IICTOS[5:0]+1)\times(32/f_{SUB})$.

Programmable Pulse Generator – PPG

The device includes a Programmable Pulse Generator, PPG, which privdes a 9-bit PPG output channel. The PPG has a programmable period of 512×T, where T is $1/f_{SYS}$ or $2/f_{SYS}$ for an output pulse width. The PPG pulse width can be limited with using the pulse width limit timer.



Note: 1. The C0INT00INT, C1VOD, C2VOD, C3VOD, C4VOD and C5VOD, are sourcd from the Comparator 0~5 outputs respectively. The INH is sourced from the Timer/Event Counter 2 output. The PPGRTG is sourced from the Timer/Event Counter 3 output.

2. The PPGOUT is internally connected to the Timer/Event Counter 2/3. The PPGDIS is internally connected to the Timer/Event Counter 3.

Programmable Pulse Generator Block Diagram

The PPG detects a trigger input and outputs a single pulse. The trigger source may come from a C0INT00INT falling edge, a Timer/Event Counter 3 retrigger mode output signal PPGRTG or a software trigger bit (PST), which can be configured by software. The PPG can output an active low pulse, active high pulse, fixed low or high setting the PPGPC register. An external pull-high or pull-low resistor is required if the PPG output is defined as an active low pulse output or an active high pulse output.

The PPG consists of a PPG control circuit, a PPGTIMER counter, a PPG counter and a PPG output control. The PPG counter consists of a 9-bit up-counter timer, two sets of 9-bit counter preload data registers and two sets of 9-bit counter approach registers. The PPG counter starts counting at the current contents in the preload register and ends at "1FFH → 000H". A "000H" data written to the PPGTA[8:0] and PPGTB[8:0] bit yields a pulse width 512×T output. Once an overflow occurs, the counter is reloaded from the PPG counter preload register, and generates a signal to stop the PPG counter. The software trigger bit (PST), will be cleared when the PPG counter overflow occurs.

Programmable Pulse Generator Registers

The overall operation of the Programmable Pulse Generator function is controlled using a series of registers.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PPGC0	PST	PRSEN	OVCEN	—	C5PSPEN	C3PSPEN	C2PSPEN	C1PSPEN
PPGC1	RLBF	C5RLEN	C3RLEN	C1RLEN	PPGE1	PPGE0	PTSYN	PCSD
PPGC2	—	—	—	DVS	PPGDEC3	PPGDEC2	PPGDEC1	PPGDEC0
PPGTA	PPGTA7	PPGTA6	PPGTA5	PPGTA4	PPGTA3	PPGTA2	PPGTA1	PPGTA0
PPGTB	PPGTB7	PPGTB6	PPGTB5	PPGTB4	PPGTB3	PPGTB2	PPGTB1	PPGTB0
PPGTC	PPGTC7	PPGTC6	PPGTC5	PPGTC4	PPGTC3	PPGTC2	PPGTC1	PPGTC0
PPGTD	PPGTD7	PPGTD6	PPGTD5	PPGTD4	PPGTD3	PPGTD2	PPGTD1	PPGTD0
PPGTEX	—	PPGTD8	—	PPGTB8	—	PPGTC8	—	PPGTA8
PWLT	D7	D6	D5	D4	D3	D2	D1	D0
PPGPC	PPGPC7	PPGPC6	PPGPC5	PPGPC4	PPGPC3	PPGPC2	PPGPC1	PPGPC0
PPGATC0	PPGSAEN	PPGSAMD	PPGSCD	PPGADJF	PPGTMMD1	PPGTMMD0	PPGACF	PPGADF
PPGATC1	PPGHTMD	—	—	PPGCNT1	PPGCNT0	PPGSA2	PPGSA1	PPGSA0
PPGATC2	—	PPGTIMES2	PPGTIMES1	PPGTIMES0	PPGACNT1	PPGACNT0	PPGASA1	PPGASA0
PPGTMC	—	—	—	PPGTON	—	—	PPGTPSC1	PPGTPSC0
PPGTMR1	D7	D6	D5	D4	D3	D2	D1	D0
PPGTMR2	D7	D6	D5	D4	D3	D2	D1	D0
PPGTMR3	D7	D6	D5	D4	D3	D2	D1	D0
PPGTMRD	D7	D6	D5	D4	D3	D2	D1	D0

Programmable Pulse Generator Register List

In the hardware automatic modify mode, the considerations listed in the following table must be taken into account when modifying the relevant bits.

C1VOD Signal	PPGSAMD	PPGSAEN	PPGTMMD[1:0]	PPGDEC[3:0]	Unchangeable Bits	Mode Description
0	x	x	xx	0000	—	—
				0001~1111	PPGTA[8:0]	Reverse oltage protection – Decrement PPG output width mode
1	0	0	xx	xxxx	—	—
	0	1	xx		PPGTA[8:0], PPGCNT[1:0], PPGSA[2:0]	Software start approach mode
	1	x	01/10		PPGTA[8:0], PPGCNT[1:0], PPGSA[2:0]	Hardware start approach mode
	1	x	00/11		—	—

"x": Don't care

• **PPGC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PST	PRSEN	OVCEN	—	C5PSPEN	C3PSPEN	C2PSPEN	C1PSPEN
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 **PST**: PPG software trigger bit
0: Stop PPG
1: Restart PPG

The PST is a software trigger bit, if this bit is set to "1" the PPG counter will start counting and this bit will be cleared to zero when a PPG counter overflow occurs or when the PPG counter stop counting. If this bit is cleared to "0", the PPG counter will stop counting.

When the PPG counter is counting and if a C0INT00INT falling edge trigger input occurs, a Timer/Event Counter 3 retrigger mode output signal PPGTRG is active or if a software control bit PST is set, the PPG counter will not be affected, that is the trigger from C0INT00INT, PPGTRG or PST will have no effect. The PST can also be used as a status bit for the PPG counter output.

Bit 6 **PRSEN**: Restart the PPG counter using C0INT00INT trigger input enable control
0: Disable
1: Enable

When restarting the PPG counter using a C0INT00INT trigger input is disabled, the PPG module output can be restarted by the software control bit, PST, only. When restarting the PPG counter using a C0INT00INT trigger input is enabled, the PPG module output can be restarted by a C0INT00INT falling edge trigger or software control by setting the PST to "1".

Note that when the valley detection function is enabled by setting the C3LEBEN to 1 and the C3PSPEN is set to 1, the PRSEN bit will not be cleared to zero by a C3VOD trigger. When the over current protection function is enabled by setting the OVCEN to 1 and the C5PSPEN is set to 1, if the PST=1, the PRSEN bit will not be cleared to zero by a C5VOD trigger.

Bit 5 **OVCEN**: Over current detection signal control
0: Disable
1: Enable

Bit 4 Unimplemented, read as "0"

Bit 3 **C5PSPEN**: Stop the PPG counter using the C5VOD falling edge trigger input enable control
0: Disable
1: Enable

When stopping the PPG counter using the C5VOD trigger input is disabled, the PPG module output can be stopped by the software control bit, PST, only. When stopping the PPG counter using the C5VOD trigger input is enabled, the PPG module output can be stopped by a C5VOD falling edge trigger or by software control by clearing the PST to "0".

Bit 2 **C3PSPEN**: Stop the PPG counter using the C3VOD falling edge trigger input enable control
0: Disable
1: Enable

When stopping the PPG counter using the C3VOD trigger input is disabled, the PPG module output can be stopped by the software control bit, PST, only. When stopping the PPG counter using the C3VOD trigger input is enabled, the PPG module output can be stopped by a C3VOD falling edge trigger or by software control by clearing the PST to "0".

Note that when the valley detection function is enabled by setting the C3LEBEN to 1 and the C3PSPEN is set to 1, the PRSEN bit will not be cleared to zero by a C3VOD trigger.

Bit 1 **C2PSPEN**: Stop the PPG counter using the C2VOD falling edge trigger input enable control
0: Disable
1: Enable
When stopping the PPG counter using the C2VOD trigger input is disabled, the PPG module output can be stopped by the software control bit, PST, only. When stopping the PPG counter using the C2VOD trigger input is enabled, the PPG module output can be stopped by a C2VOD falling edge trigger or by software control by clearing the PST to "0".

Bit 0 **C1PSPEN**: Stop the PPG counter using the C1VOD falling edge trigger input enable control
0: Disable
1: Enable
When stopping the PPG counter using the C1VOD trigger input is disabled, the PPG module output can be stopped by the software control bit, PST, only. When stopping the PPG counter using the C1VOD trigger input is enabled, the PPG module output can be stopped by a C1VOD falling edge trigger or by software control by clearing the PST to "0".

• PPGC1 Register

Bit	7	6	5	4	3	2	1	0
Name	RLBF	C5RLEN	C3RLEN	C1RLEN	PPGE1	PPGE0	PTSYN	PCSD
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 **RLBF**: PPG counter reload control bit
0: From PPGTA[8:0]
1: From PPGTB[8:0]
The PPG counter will be reloaded by one of the conditions which include a PPG counter overflow, PPG off, any action causing the PPG to stop.
Normally, if RLBF=0, the PPG counter is reloaded from the preload register A. If C1RLEN=1, C3RLEN=1 or C5RLEN=1, when a C1VOD, C3VOD or C5VOD falling edge occurs, the RLBF will be set to "1" and the PPG counter will be reloaded from preload register B until the RLBF is cleared by software.

Bit 6 **C5RLEN**: C5VOD falling edge to set RLBF for PPG counter being reloaded from preload register B enable control
0: Disable
1: Enable

Bit 5 **C3RLEN**: C3VOD falling edge to set RLBF for PPG counter being reloaded from preload register B enable control
0: Disable
1: Enable

Bit 4 **C1RLEN**: C1OD falling edge to set RLBF for PPG counter being reloaded from preload register B enable control
0: Disable
1: Enable

Bit 3~2 **PPGE1~PPGE0**: PPG enable control
01: Enable – PPG can be restarted by hardware or software
Others: Disable – PPG keeps always inactive and ignores any restart trigger
Note that when PPGE[1:0]≠01, PST bit is fixed to 0, the writing operation is invalid.

Bit 1 **PTSYN**: PPG start counting synchronised with clock or not
0: Synchronised with clock
1: Asynchronised with clock

Bit 0 **PCSD**: PPG counter and pulse width limiter timer clock source, f_{PPG}, selection
0: f_{SYS}
1: $f_{SYS}/2$

• **PPGC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	DVS	PPGDEC3	PPGDEC2	PPGDEC1	PPGDEC0
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 **DVS**: PPGTA reverse voltage increment period selection
0: Every $8/f_{SYS}$
1: Every $16/f_{SYS}$
When the C1VOD signal is low and PPGDEC[3:0]≠0000, the PPGTA is automatically increased one time, then the PPGTA will be automatically increased by a specific increment value every $8/f_{SYS}$ or $16/f_{SYS}$, the increment value depends on the PPGDEC[3:0] bits. When the C1VOD signal is high, the PPGTA will not automatically increase.

Bit 3~0 **PPGDEC3~PPGDEC0**: PPGTA automatic increment value selection
0000: 0
0001: 1
0010: 2
0011: 3
0100: 4
0101: 5
0110: 6
0111: 7
1000: 8
1001: 9
1010: 10
1011: 11
1100: 12
1101: 13
1110: 14
1111: 15

Note: When using the PPG function, the most important point to note is to ensure that the CMP1 settings and C1VOD signal set high before setting the PPGC2 register. Since the C1VOD signal state is unknown, if PPGDEC[3:0]≠0000, the PPGTA[8:0] value will be automatically incremented by a specific value every $8/f_{SYS}$ or $16/f_{SYS}$ until it is incremented to 1FFH. The incremented value depends on the PPGDEC[3:0] bits.

• **PPGATC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PPGSAEN	PPGSAMD	PPGSCD	PPGADJF	PPGTMMD1	PPGTMMD0	PPGACF	PPGADF
R/W	R/W	R/W	R/W	R	R	R	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PPGSAEN**: PPGTA approach mode enable control
0: Disable
1: Enable

This bit is only valid when PPGSAMD=0. Since the pulse width approach operation is controlled by hardware when PPGSAMD=1, a software write is invalid.

PPGSAMD	PPGSAEN		
	Software Write	Hardware Write	Read
0	0	x	0
	1		1
1	0	0	0
	0	1	1
	1	0	0
	1	1	1

"x": Don't care

Bit 6 **PPGSAMD**: PPGTA approach mode selection
0: Software approach mode
1: Hardware approach mode
The PPGTON bit will be cleared to zero and the PPGTIMER counter will reload the PPGTMR1 register value if this bit changes from 0 to 1.

Bit 5 **PPGSCD**: PPGTA approach bits selection
0: PPGTC[8:0]
1: PPGTD[8:0]
This bit is only valid when PPGSAMD=0.

Bit 4 **PPGADJF**: PPG register modification flag
0: PPG related registers can be changed
1: PPG related registers cannot be changed
If this bit set to high, the PPGTA[8:0] bits in the PPGTEX and PPGTA registers, the PPGCNT[1:0] and PPGSA[2:0] bits in the PPGATC1 register cannot be changed by software.

Bit 3~2 **PPGTMMD1~PPGTMMD0**: PPGTIMER operating mode
00: PPGTA floating mode (t0~t1 interval)
01: PPGTA approach PPGTC mode (t1~t2 interval)
10: PPGTA approach PPGTD mode (t2~t3 interval)
11: PPGTA floating mode (t3~t0 interval)
These bits are only valid when PPGSAMD=1.

Bit 1 **PPGACF**: PPGTA approach PPGTC operation complete flag
0: PPGTA approach PPGTC operation has not completed
1: PPGTA approach PPGTC operation has completed
This bit can be cleared to zero by software, but it cannot be set high by software. If this bit is high, it also can be automatically cleared to zero by the hardware when PPGSAMD=0 and PPGSAEN=1; or if PPGSAMD=1 and PPGHTMD=0, when a C4VOD rising trigger occurs; or if PPGSAMD=1 and PPGHTMD=1, when the PPGTON bit changes from 0 to 1.

Bit 0 **PPGADF**: PPGTA approach PPGTD operation complete flag
0: PPGTA approach PPGTD operation has not completed
1: PPGTA approach PPGTD operation has completed
This bit can be cleared to zero by software, but it cannot be set high by software. If this bit is high, it also can be automatically cleared to zero by the hardware when PPGSAMD=0 and PPGSAEN=1; or if PPGSAMD=1 and PPGHTMD=0, when a C4VOD rising trigger occurs; or if PPGSAMD=1 and PPGHTMD=1, when the PPGTON bit changes from 0 to 1.

• **PPGATC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PPGHTMD	—	—	PPGCNT1	PPGCNT0	PPGSA2	PPGSA1	PPGSA0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 **PPGHTMD**: PPGTIMER counter trigger source selection in the hardware approach mode
0: C4VOD rising
1: PPGTON

Bit 6~5 Unimplemented, read as "0"

Bit 4~3 **PPGCNT1~PPGCNT0**: PPG trigger times selection (Variable: M)
00: 1
01: 2
10: 3
11: 4

Bit 2~0 **PPGSA2~PPGSA0**: PPGTA approach value selection (Variable: N)
000: ±1
001: ±2
010: ±3
011: ±4
100: ±5
101: ±6
110: ±7
111: ±8

• **PPGATC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PPGTIMES2	PPGTIMES1	PPGTIMES0	PPGACNT1	PPGACNT0	PPGASA1	PPGASA0
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 6~4 **PPGTIMES2~PPGTIMES0**: Approach times selection – change M and N values
000: 1
001: 2
010: 3
011: 4
100: 5
101: 6
110: 7
111: 8

Bit 3~2 **PPGACNT1~PPGACNT0**: PPG trigger times change selection – change M value
00: Unchanged
01: Unchanged
10: +1
11: -1

Note: 1. When PPGCNT[1:0]=00/11, the PPGCNT[1:0] bits will not increase or decrease according to PPGACNT[1:0] and will be fixed at 00 or 11.
2. In the hardware approach mode, when PPGACNT[1:0]=10, it is increased by 1 in the t1~t2 interval and decreased by 1 in the t2~t3 interval. When PPGACNT[1:0]=11, it is decreased by 1 in the t1~t2 interval and increased by 1 in the t2~t3 interval.
3. If the PPGTA has reached the approach mode, PPGACNT[1:0] is unchanged.

Bit 1~0 **PPGASA1~PPGASA0**: PPGTA approach value change selection – change N value
00: Unchanged
01: Unchanged
10: +1
11: -1

Note: 1. When PPGSA[2:0]=000/111, the PPGSA[2:0] bits will not increase or decrease according to PPGASA[1:0] and will be fixed to at 000 or 111.
2. In the hardware approach mode, when PPGASA[1:0]=10, it is increased by 1 in the t1~t2 interval and decreased by 1 in the t2~t3 interval. When PPGASA[1:0]=11, it is decreased by 1 in the t1~t2 interval and increased by 1 in the t2~t3 interval.

• PPGTA Register

Bit	7	6	5	4	3	2	1	0
Name	PPGTA7	PPGTA6	PPGTA5	PPGTA4	PPGTA3	PPGTA2	PPGTA1	PPGTA0
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 **PPGTA7~PPGTA0**: PPG counter preload register A bit 7 ~ bit 0

• PPGTB Register

Bit	7	6	5	4	3	2	1	0
Name	PPGTB7	PPGTB6	PPGTB5	PPGTB4	PPGTB3	PPGTB2	PPGTB1	PPGTB0
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 **PPGTB7~PPGTB0**: PPG counter preload register B bit 7 ~ bit 0

• PPGTC Register

Bit	7	6	5	4	3	2	1	0
Name	PPGTC7	PPGTC6	PPGTC5	PPGTC4	PPGTC3	PPGTC2	PPGTC1	PPGTC0
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 **PPGTC7~PPGTC0**: PPG counter approach register C bit 7 ~ bit 0

• PPGTD Register

Bit	7	6	5	4	3	2	1	0
Name	PPGTD7	PPGTD6	PPGTD5	PPGTD4	PPGTD3	PPGTD2	PPGTD1	PPGTD0
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 **PPGTD7~PPGTD0**: PPG counter approach register D bit 7 ~ bit 0

• **PPGTEX Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PPGTD8	—	PPGTB8	—	PPGTC8	—	PPGTA8
R/W	—	R/W	—	R/W	—	R/W	—	R/W
POR	—	x	—	x	—	x	—	x

"x": Unknown

Bit 7 Unimplemented, read as "0"
Bit 6 **PPGTD8**: PPG counter approach register D bit 8
Bit 5 Unimplemented, read as "0"
Bit 4 **PPGTB8**: PPG counter preload register B bit 8
Bit 3 Unimplemented, read as "0"
Bit 2 **PPGTC8**: PPG counter approach register C bit 8
Bit 1 Unimplemented, read as "0"
Bit 0 **PPGTA8**: PPG counter preload register A bit 8

• **PWLT 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**: PPG pulse width limit timer bit 7~bit 0
The pulse width limit is (256-PWLT)/(f_{PPG}/2).

• **PPGPC Register**

Bit	7	6	5	4	3	2	1	0
Name	PPGPC7	PPGPC6	PPGPC5	PPGPC4	PPGPC3	PPGPC2	PPGPC1	PPGPC0
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 **PPGPC7~PPGPC0**: PPG output control
01010101: PPG output is active low, inactive floating
10101010: PPG output is active high, inactive floating
10001101: Reserved
00110011: PPG output is forced to high
00110010: PPG output is forced to low
Other values: PPG output is floating

• **PPGTMC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	PPGTON	—	—	PPGTPSC1	PPGTPSC0
R/W	—	—	—	R/W	—	—	R/W	R/W
POR	—	—	—	0	—	—	0	0

Bit 7~5 Unimplemented, read as "0"
Bit 4 **PPGTON**: PPGTIMER counting enable control
0: Disable
1: Enable
Writing PPGTON is invalid when PPGSAMD=1 and PPGHTMD=0.
Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **PPGTPSC1~PPGTPSC0**: PPGTIMER prescaler rate selection
00: $f_H/128$
01: $f_H/256$
10: $f_H/512$
11: $f_H/1024$

• **PPGTMR1 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**: PPGTIMER preload register T1 bit 7 ~ bit 0

• **PPGTMR2 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**: PPGTIMER preload register T2 bit 7 ~ bit 0

• **PPGTMR3 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**: PPGTIMER preload register T3 bit 7 ~ bit 0

• **PPGTMRD 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**: PPGTIMER register bit 7 ~ bit 0

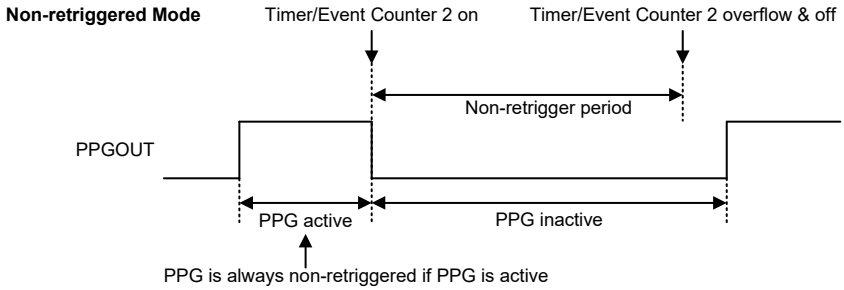
Writing Data to PPGTA~PPGTD & PPGTEX Register Description

When writing data to the PPGTA~PPGTD & PPGTEX registers, users need to write the high byte first after which the low byte can be written. This means that the PPGTA8/PPGTB8/PPGTC8/PPGTD8 bit in the PPGTEX register must be written first, after which the PPGTA[7:0]/PPGTB[7:0]/PPGTC[7:0]/PPGTD[7:0] bits in the corresponding register can be written. The register contents do not take effect until the low byte has been written. If the value of the PPGTEX register is updated, and data is written to the PPGTA register only, the PPGTB8, PPGTC8 and PPGTD8 bits in the PPGTEX register will not be updated. When reading the PPGTEX register, only the PPGTA8 bit will have the updated value, the PPGTB8, PPGTC8 and PPGTD8 bits will retain their previous written values.

Non-retrigger Function

The PPG unit has a non-retrigger function to inhibit further PPG triggers. The PPG will be non-triggered by one of the following conditions:

- PPG is active
- During the non-retrigger period which starts counting once the PPG has stopped. Only available when used with PPG non-retrigger function mode of the Timer/Event Counter 2, the non-trigger period is determined by the Timer/Event Counter 2 which will start counting when the PPG output active to inactive transition occurs.



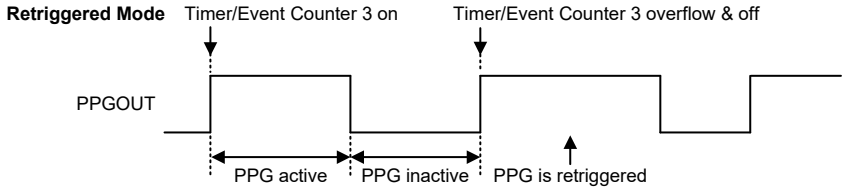
Note: 1. If T2ON=1, when the INH signal is high, the PPG non-retrigger mode will be enabled to inhibit further PPG triggers until the Timer/Event Counter 2 overflows or the T2ON bit is cleared to zero. When the INH signal is low, the PPG can be triggered again and the signal can be output normally.

2. During the non-retrigger period, the PPG cannot be triggered by the C0INT00INT trigger signal, but the PPG module can be triggered by the software control bit PST.

Retrigger Function

The PPG unit has a retrigger function for further PPG trigger. The PPG will be retriggered by one of the following conditions:

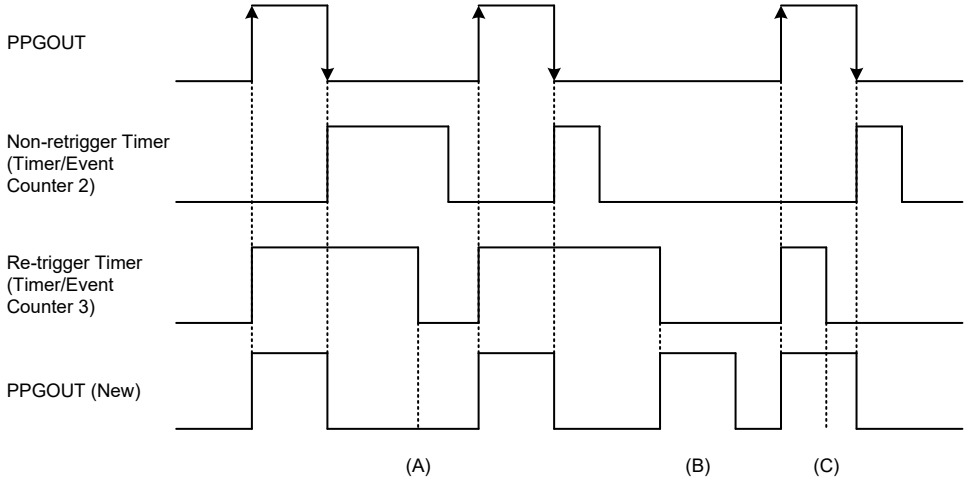
- PPG is active.
- During the retrigger period which starts counting once the PPG has started. Only available when used with PPG retrigger function mode of the Timer/Event Counter 3.



The PPG non-retrigger and retrigger signals are shown in the following table.

Timer/Event Counter n	TnM[1:0]	TnRSEN	T2ON/ T3ON	Overflow or not	INH/ PPGRTG	Description
n=2	00 (Non-retrigger Mode)	x	1	No overflow	INH=1	Inhibit PPG retrigger
				Overflow	INH=0	PPG can be retriggered
n=3	00 (Retrigger Mode)	1	1	No overflow	PPGRTG=0	PPG has not retriggered
				Overflow	PPGRTG=1	PPG retrigger

The PPG output waveforms, after the non-retrigger and retrigger functions are used, are shown below.



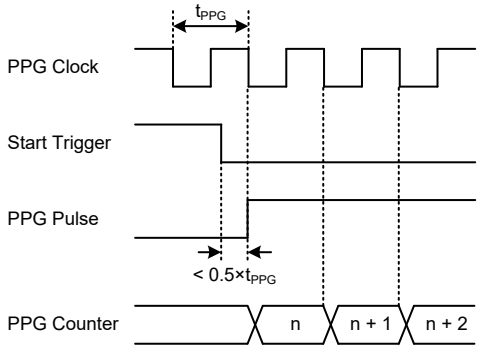
(A) PPG is not triggered because it is in the non-triggered period.
(B) PPG is triggered because it is in the retriggered period.
(C) When PPG is active, it cannot be triggered again.

PPG Synchronised with Clock Description

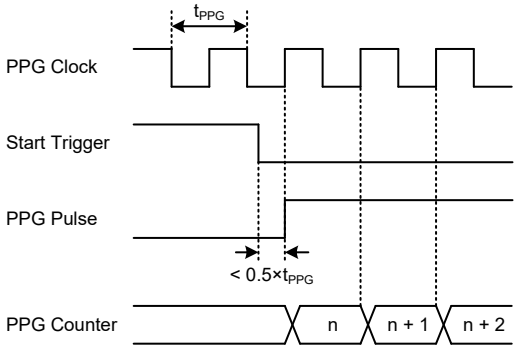
When the PPG counter starts counting and whether it is synchronised with the clock or not is determined by the PTSYN bit in the PPGC0 register.

To control the PPG pulse starting delay is less than or equal to $0.5\times(1/f_{PPG})$ when start synchronised with clock is selected, the f_{PPG} clock rising edge or falling edge, which triggers the PPG, varies with next coming clock transition once the PPG starts. After the PPG starts, the PPG output becomes active and begins to count as soon as first transition the falling or rising of f_{PPG} clock arrives. After the first trigger has completed, the following clock edge trigger type is determined by the first one. For example, once the PPG starts and the following clock transition is a falling edge, the PPG will be triggered by a falling edge until the PPG stops and vice versa.

Example 1: Since the first trigger type is a falling edge after the PPG starts, the PPG counter is triggered by a falling edge until the PPG stops.



Example 2: Since the first trigger type is a rising edge after the PPG starts, the PPG counter is triggered by a rising edge until the PPG stops.



To Start the PPG Operation

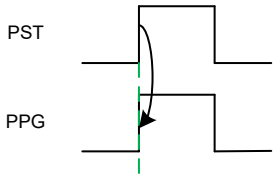
- Set the PPG output status using the PPGPC register
- Determine whether the PPG timer start counting is synchronised with the f_{PPG} clock or not using the PTSYN bit
- Set the PPG input mode using the PRSEN and CnPSPEN bits in the PPGC0 register (n=1~3, 5)
- Set the PPG output pulse width by writing data to the PPGTA, PPGTB and PPGTEX registers
- Determine whether to use the CnVOD falling edge to enable the reload function from preload register B or not by setting the CnRLEN bit in the PPGC0 register (n=1/3/5)
- Determine whether to use the non-retrigger period function or not using the PPG non-retrigger function mode of Timer/Event Counter 2
- Determine whether to use the retrigger period function or not using the PPG retrigger function mode of Timer/Event Counter 3
- Set the pulse width limit timer for the pulse width limit function using the PWLT register
- Set the PPG clock by PCSD bit
- Set the PPGE[1:0] bits to enable PPG
- When the PPG is triggered by a C0INT00INT falling edge, a Timer/Event Counter 3 retrigger mode output signal PPGRTG or triggered by a software trigger bit, PST, being set to “1”, the PPG will start counting from the current value of the preload register. If a PPG counter overflow occurs, a pulse width limit condition occurs or the PPG is trigged by a software bit, PST, being cleared to zero or by a CnVOD (n=1~3, 5) falling edge, the PPG will stop counting.

The Methods to Start PPG Output

The methods to start PPG output, which can be divided into software starting and hardware starting. There are three kinds of hardware starting signals, namely synchronization signal starting signal (C0INT00INT) and TM timer timing start signal (PPGRTG). As shown below.

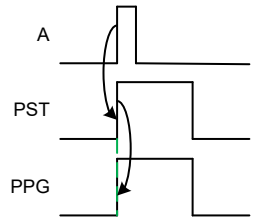
Software Start

- After the PST bit is set to 1 by software, PPG starts to output, the waveform figure is shown below



Hardware Start

- When receives a trigger signal (A), the PST bit will be set to 1, PPG starts to output, the waveform figure is shown below



Note: In the figure, “A” represents C0INT00INT signal and PPGRTG signal.

Note: 1. If the C0INT00INT signal is used to start PPG output, the PRSEN bit should be set to 1 to enable the C0INT00INT and trigger the PST bit to start PPG output.

2. If the PPGRTG signal is used to start PPG output, the T3RSEN bit should be set to 1 to enable the PPGRTG and trigger the PST bit to start PPG output.

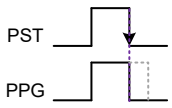
To Stop the PPG Function

The PPG can stop when a PPG timer overflow occurs, a CnVOD falling edge occurs (CnPSPEN=1, n=1/2/3/5), a software stop condition (PST=1→0) occurs or the pulse width limit is reached. Any action causing the PPG to stop will generate the following actions:

- PPG timer will be reloaded
- PST bit is cleared to zero
- PPG will be inactive

Software Stop

After the PST bit is cleared by the software, PPG stops output, as shown in the following figure.



Hardware Stop

The PPG hardware stop is implemented by CnVOD(n=1/2/3/5) falling edge trigger together with the CnPSPEN bits.

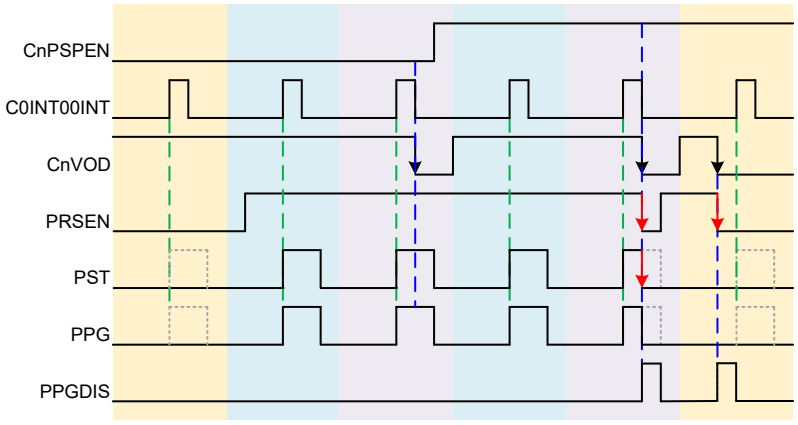
The CnPSPEN bits are the PPG stopping enable or disable bits using the CnVOD(n=1/2/3/5) trigger input. If these bits are enabled, the PPG timer stopping input can be triggered by a CnVOD(n=1/2/3/5) falling edge.

The PRSEN bit will be cleared to zero by a CnVOD(n=1/2/3/5) falling edge, no matter whether the PPG is in an active period or not. This will prevent the PPG module output from being restarted by a C0INT00INT falling edge occurring again, it can only be restarted by software when PRSEN is set again by software.

- n=1&2:

CnVOD	CnPSPEN	PST	PRSEN	PPGDIS	PPG
0/↑/1/↓	0	u	u	0	u
0/↑/1	1	u	u	0	u
↓		0	0	1	floating

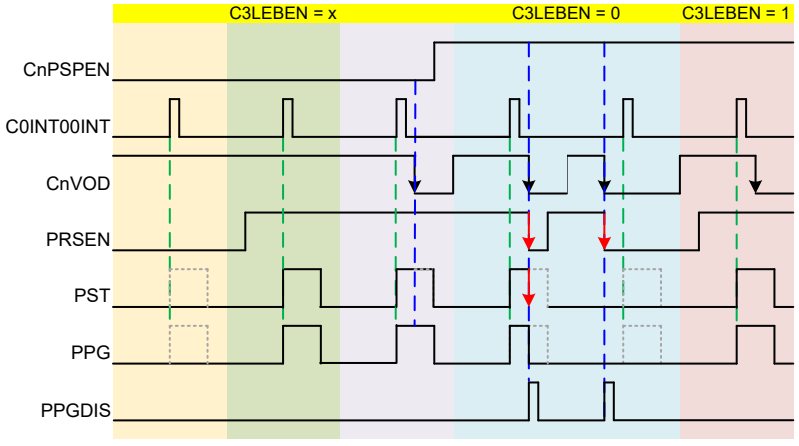
"u": unchanged



- n=3:

CnVOD	CnPSPEN	C3LEBEN	PST	PRSEN	PPGDIS	PPG
0/↑/1/↓	0	x	u	u	0	u
0/↑/1/	1	x	u	u	0	u
↓		0	0→0	u→0	1	Floating
			1→0	u→0	1	Floating
		1	u	u	0	u

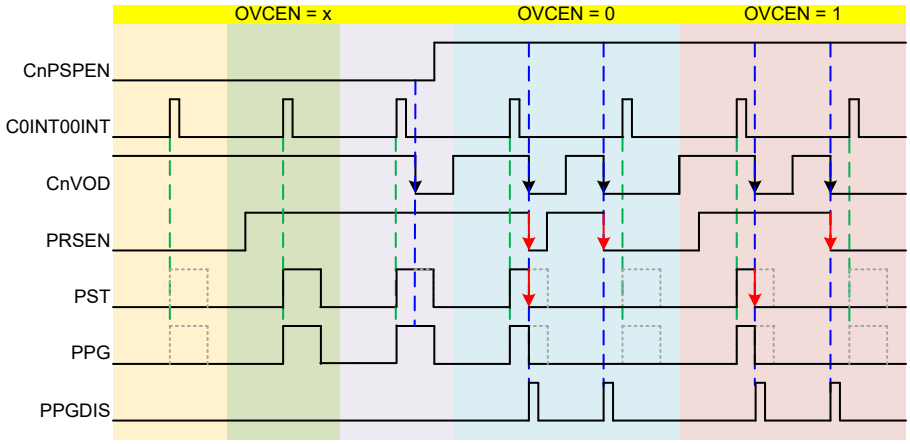
"x": Don't care; "u": Unchanged



- n=5:

CnVOD	CnPSPEN	OVCEN	PST	PRSEN	PPGDIS	PPG
0/↑/1/↓	0	x	u	u	0	u
0/↑/1/	1	x	u	u	0	u
↓		0	0→0	u→0	1	floating
			1→0		1	floating
		1	0→0	u→0	1	floating
			1→0	u	1	floating

"x": Don't care; "u": Unchanged



Pulse Width Limit Function

The PPG unit has a pulse width limit function to stop the PPG output. The PPG output will be stopped once the pulse width has reached the limit. This function is implemented by a pulse width limit timer which starts counting once the PPG is triggered and stops once it overflows or then the PPG is stopped. The pulse width limit is (256-PWLT)/(f_{PPG}/2), where PWLT is the value in the pulse width limit timer register, PWLT. Note that the pulse width limit timer may have an error of about f_{PPG}/2.

To reload the pulse width limit function:

- Pulse width limit timer overflow
- PPG is triggered

PPG Load Register Function

The PPG can select load register by software setting (RLBF bit), or by C1VOD, C3VOD, or C5VOD falling edge trigger (C1RLEN/C3RLEN/C5RLEN=1).

Software Setting

When the RLBF bit is set by the software, it is necessary to select the PPG counter loading value, as shown in the following table.

RLBF	PPG Load Register
0	PPGTA
1	PPGTB

Hardware Setting

Set the CnRLEN(n=1/3/5) bit to "1". When the CnVOD signal changes from high to low, it will generate a falling edge trigger signal and the PPG output signal will be changed from the original PPGTA to PPGTB.

CnVOD	CnRLEN	RLBF	PPG Load Register
0/↑/1/↓	0	0	PPGTA
0/↑/1/↓		1	PPGTB
0/↑/1	1	0	PPGTA
↓		0 → 1	PPGTA → PPGTB
0/↑/1/↓		1	PPGTB

Reverse Voltage Protection Adjustment Function

When C1VOD="↓", the PPGTA will increase the value set by PPGDEC[3:0] and starts the DVS counting. When the C1VOD=0 and the DVS counting time has reached, the PPGTA will increase the value set by PPGDEC[3:0] again. Stop adjusting until PPGTA=1FFH or C1VOD=1.

For example, when DVS=1(16×t_{SYS}), PPGDEC[3:0]=1000(8), PPGTA[8:0]=1 0000 0000(256).

C1VOD	DVS	PPGTA[8:0]
1	Disable	256
↓	Disable → Enable	256+8=264
0	16×t_{SYS}	264+8=272
0	32×t_{SYS}	272+8=280
0	48×t_{SYS}	280+8=288
↑	62×t_{SYS} → Disable	288
1	Disable	288

PPGTA Approach Function

The PPGTA approach function can only operate when C1VOD=1, that is, no reverse voltage occurs. When the PPG is operating in the approach mode, the PPG will immediately operate in the PPG reverse voltage protection adjustment mode once C1VOD=0. The PPGTA approach mode has both software control and hardware control. The differences and setup steps are described below.

Software Approach Mode – PPGSAMD=0

Users can select when to start the PPGTA approach function, PPGTA approach PPGTC or PPGTD. When PPGSAEN=1, the PPGADJF bit will also be set to high by the hardware. At this time, the PPGTA register, the PPGCNT[1:0] bits in the PPGATC1 register and the PPGSA[2: 0] bits in the PPGATC1 register must not be changed by software, the PPGACF and PPGADF bits can also be cleared to zero by hardware until PPGTA=PPGTC/PPGTD, and their corresponding flags will be set high.

In the software approach mode, the PPGTIMER counter operates in the general timer mode and the counting value is loaded by PPGTMR1 register. When PPGTON=1, the PPGTIMER counter counts from the PPGTMR1 register, if the counter overflow will trigger the PPGTMINT signal.

The following summarises the individual steps that should be executed in order to implement a PPGTA approach process in the software approach mode.

Write the initial value to the PPGTA~PPGTD & PPGTEX registers. Note that the high byte, PPGTEX, needs to be written first after which the low byte, PPGTA~PPGTD, can be written to ensure the PPGTA[8:0]~PPGTD[8:0] bits will be correctly written.

Step 1. Set the PPG trigger times and the approach value by configuring the PPGACNT[1:0] and PPGASA[2:0] bits the PPGATC1 register respectively.

Step 2. Select after how many times to adjust the PPG trigger times and the approach value by setting the PPGTIMES[2:0] bits in PPGATC2 register.

Step 3. Select how to change the PPG trigger times by setting the PPGACNT[1:0] bits in the PPGATC2 register.

Step 4. Select how to change the approach value by setting the PPGASA[1:0] bits in PPGATC2 register.

Step 5. Clear the PPGSAMD bit in the PPGATC0 register to zero.

Step 6. Select whether the PPGTA approaches PPGTC or PPGTD by setting the PPGSCD bit in the PPGATC0 register.

Step 7. Setup other PPG related registers. Refer to the Programmable Pulse Generator Registers for details.

Step 8. If the PPGATCD interrupt is used, the interrupt control registers must be correctly configured to ensure the PPGATCD interrupt function is active. The master interrupt control bit, EMI, the PPGATCD interrupt control bit, PPGATCDE, and associated Multi-function interrupt enable bit must be set high in advance.

Step 9. Set the PPGSAEN bit in the PPGATC0 register to 1 to enable the PPGTA approach function.

Step 10. Read the PPGACF and PPGADF bits to determine whether PPGTA is equal to PPGTC or PPGTD.

Note: 1. If the method of polling the PPGACF and PPGADF bits is used, the interrupt enable step above can be omitted.

2. After entering the interrupt vector, it is necessary to read whether PPGACF or PPGADF is set to “1” to ensure that the setting value is approached correctly.

Hardware Approach Mode – PPGSAMD=1

In the hardware approach mode, how the PPGTA changes depends on the PPGTIMER timing interval. The PPGTIMER counter has two trigger signals which are selected by the PPGHTMD bit. The PPGTIMER counter will be started if an active C4VOD rising edge trigger source is occurred or the PPGTON bit changes from 0 to 1 by software. The PPG will execute different actions in four time intervals. The t0~t1 interval is when the PPGTIMER counter counts from the PPGTMR1 value to the timer overflow. The PPG will output the same pulse width, that is, the PPGTA[8:0] bits values are fixed and will not automatically adjust. The t1~t2 interval is when the PPGTIMER counter counts from the PPGTMR2 value to the timer overflow, the PPGTA will approach PPGTC according to the PPGCNT[1:0] and PPGSA[2:0] bits value. If the approach time is reached, which is setup by the PPGTIMES[2:0], the PPGCNT[1:0] and PPGSA[2:0] bits will change according to the PPGACNT[1:0] and PPGASA[1:0] bits. The PPGTA value will remain unchanged until PPGTA is equal to PPGTC or the timer overflows. The t2~t3 interval is when the PPGTIMER counts from the PPGTMR3 value to the timer overflows, PPGTA will approach PPGTD according to the PPGCNT[1:0] and PPGSA[2:0] bits. If the approach time is reached, which is setup by the PPGTIMES[2:0], the PPGCNT[1:0] and PPGSA[2:0] bits will change according to the PPGACNT[1:0] and PPGASA[1:0] settings. The PPGTA value will remain unchanged until PPGTA is equal to PPGTD or the timer overflows. The t3~t0 interval is when the PPGTIMER disabled, where the PPG related parameters can be change by software.

Note that before t1 occurs, the PPG related registers must be setup. Otherwise, the PPGTA[8:0], PPGCNT[1:0] and PPGSA[2:0] bits cannot be changed in the t1~t2 interval. These bits cannot be changed until t3 occurs.

In the t1~t2 interval, when PPGTA is equal to PPGTC, the PPGACF bit will be set to 1 by the hardware. Note that the PPGACF bit can be cleared by the software, but it cannot set high by the software. When PPGACF=1, the software does not clear this bit to zero, it also can be automatically clear to zero by the hardware when the next C4VOD rising edge or PPGTON bit changes from 0 to 1 occurs.

In the t2~t3 interval, when PPGTA is equal to PPGTD, the PPGADF bit will be set to 1 by the hardware. Note that the PPGADF bit can be cleared by the software, but it cannot set high by the software. When PPGADF=1, the software does not clear this bit, it also can be automatically clear to zero by the hardware when the next C4VOD rising edge occurs or when the PPGTON changes from 0 to 1.

In the hardware approach mode, if users want to stop the related function, the PPG can be changed to the software mode by clearing the PPGSAMD bit to zero.

C1VOD Signal	PPGSAMD	PPGSAEN	PPGSCD	PPGTMMD[1:0]	Description
1	0	0	x	xx	The PPGTA[8:0] value does not update automatically.
1	0	1	0	xx	The PPGTA[8:0] approaches the PPGTC[8:0] times according to the PPGCNT[1:0] bits, the approach value is determined by the PPGSA[2:0] bits.
1	0	1	1	xx	The PPGTA[8:0] approaches the PPGTD[8:0] times according to the PPGCNT[1:0] bits, the approach value is determined by the PPGSA[2:0] bits.
1	1	0	x	00	The PPGTA[8:0] value does not update automatically.
1	1	1 (by hardware)	x	01	The PPGTA[8:0] approaches the PPGTC[8:0] times according to the PPGCNT[1:0] bits, the approach value is determined by the PPGSA[2:0] bits.
1	1	1 (by hardware)	x	10	The PPGTA[8:0] approaches the PPGTD[8:0] times according to the PPGCNT[1:0] bits, the approach value is determined by the PPGSA[2:0] bits.
1	1	0	x	11	The PPGTA[8:0] value does not update automatically.

"x": Don't care

The following summarises the individual steps that should be executed in order to implement a PPGTA approaching process in the hardware approach Mode.

Write the initial value to the PPGTA~PPGTD & PPGTEX registers. Note that the high byte, PPGTEX, needs to be written first after which the low byte, PPGTA~PPGTD, can be written to ensure the PPGTA[8:0]~PPGTD[8:0] bits will be correctly written.

Step 1. Set the PPG trigger times and the approach value by configuring the PPGACNT[1:0] and PPGASA[2:0] bits in the PPGATC1 register.

Step 2. Select the hardware trigger source by setting the PPGHTMD bit in the PPGATC1 register.

Step 3. Select after how many times to adjust the PPG trigger times and the approach value by setting the PPGTIMES[2:0] bits in the PPGATC2 register.

Step 4. Select how to change the PPG trigger times by setting the PPGACNT[1:0] bits in the PPGATC2 register.

Step 5. Select how to change the approach value by setting the PPGASA[1:0] bits in the PPGATC2 register.

Step 6. Set the PPGTMR1, PPGTMR2 and PPGTMR3 counter values.

Step 7. Select the PPGTIMER clock source by setting the PPGTPSC[1:0] bits in the PPGTMC register.

Step 8. Setup other PPG related registers. Refer to the Programmable Pulse Generator Registers for details.

Step 9. If the PPGATCD interrupt is used, the interrupt control registers must be correctly configured to ensure the PPGATCD interrupt function is active. The master interrupt control bit, EMI, the PPGATCD interrupt control bit, PPGATCDE, and associated Multi-function interrupt enable bit must be set high in advance.

Step 10. Set the PPGSAMD bit in the PPGATC0 register to 1. Note that the PPGTON bit will be cleared to zero by hardware. If the PPGHTMD bit is 0, once an active C4VOD rising edge trigger source occurrence will trigger a hardware action, it is important to ensure that other relevant settings are completed before setting this bit to avoid unpredictable errors.

Step 11. Determine whether PPGTA is equal to PPGTC or PPGTD by reading the PPGACF and PPGADF bits.

Step 12. Read the PPGTMMD[1:0] bits to determine the PPGTIMER current operating mode.

Note: 1. If the method of polling the PPGACF and PPGADF bits is used, the interrupt enable step above can be omitted.
2. After entering the interrupt vector, it is necessary to read whether PPGACF or PPGADF is set to "1" to ensure that the setting value is approached correctly.

Example:

1. PPGSAMD=1; PPGTPSC0=1; PPGHTMD=0
2. PPGCNT[1:0]=10B; PPGSA[2:0]=011B
   This means that PPGTA is increased by 4 for every 3 PPG triggers.
3. PPGTIMES[2:0]=001B; PPGACNT[1:0]=10B; PPGASA[1:0]=00B
   This means that the PPG trigger times is increased by one and the increment value remains unchanged for every 2 PPG triggers. Therefore PPGTA is increased by 4 for every 4 PPG triggers.
4. PPGTA=400; PPGTC=420; PPGTD=410

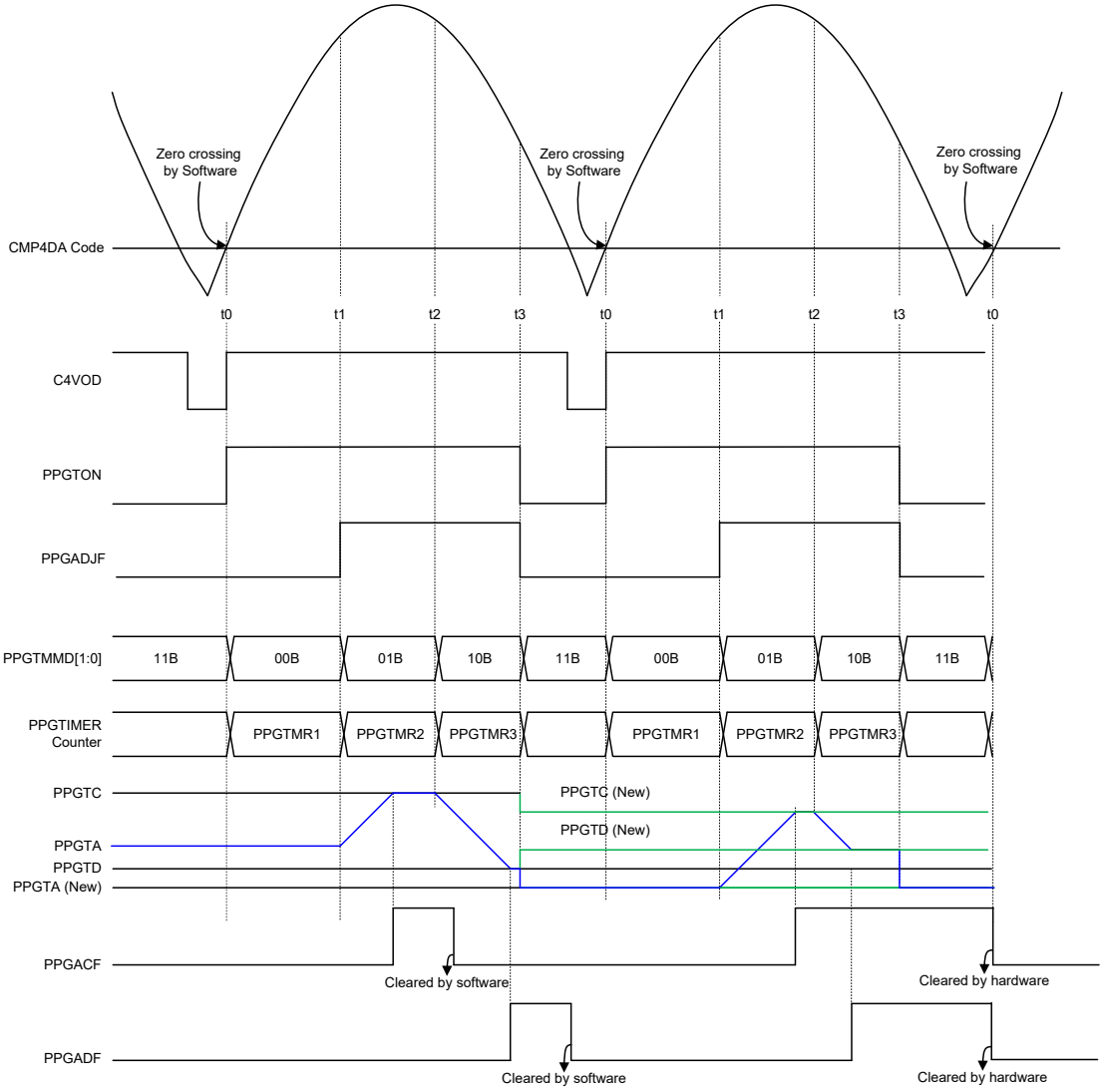
Signals/ Bits	C0INT00INT	C4VOD	PPGSAEN	PPGADJF	PPGTMMD[1:0]	PPGTON	PPGTA[8:0]	PPGTC[8:0]	PPGTD[8:0]	PPGACF	PPGADF
~t0		0 / 1 / ↓	0	0	11	0	400	420	410	0/1	0/1
t0~t1		↑	0	0	00	1	400	420	410	0	0
t1~t2	↓	1	1	1	01	1	400	420	410	0	0
	↓	1	1	1	01	1	400	420	410	0	0
	↓	1	1	1	01	1	404	420	410	0	0
	↓	1	1	1	01	1	404	420	410	0	0
	↓	1	1	1	01	1	404	420	410	0	0
	↓	1	1	1	01	1	408	420	410	0	0
	↓	1	1	1	01	1	408	420	410	0	0
	↓	1	1	1	01	1	408	420	410	0	0
	↓	1	1	1	01	1	408	420	410	0	0
	↓	1	1	1	01	1	412	420	410	0	0
	↓	1	1	1	01	1	412	420	410	0	0
	↓	1	1	1	01	1	412	420	410	0	0
	↓	1	1	1	01	1	412	420	410	0	0
	↓	1	1	1	01	1	416	420	410	0	0
	↓	1	1	1	01	1	416	420	410	0	0
	↓	1	1	1	01	1	416	420	410	0	0
	↓	1	1	1	01	1	416	420	410	0	0
	↓	1	1	1	01	1	416	420	410	0	0
	↓	1	1	1	01	1	420	420	410	1	0

Signals/ Bits	C0INT00INT	C4VOD	PPGSAEN	PPGADJF	PPGTMMD[1:0]	PPGTON	PPGTA[8:0]	PPGTC[8:0]	PPGTD[8:0]	PPGACF	PPGADF
t2~t3	↓	1	1	1	10	1	420	420	410	1	0
	↓	1	1	1	10	1	420	420	410	1	0
	↓	1	1	1	10	1	420	420	410	1	0
	↓	1	1	1	10	1	420	420	410	1	0
	↓	1	1	1	10	1	416	420	410	1	0
	↓	1	1	1	10	1	416	420	410	1	0
	↓	1	1	1	10	1	416	420	410	1	0
	↓	1	1	1	10	1	416	420	410	1	0
	↓	1	1	1	10	1	416	420	410	1	0
	↓	1	1	1	10	1	412	420	410	1	0
	↓	1	1	1	10	1	412	420	410	1	0
	↓	1	1	1	10	1	412	420	410	1	0
	↓	1	1	1	10	1	412	420	410	1	0
	↓	1	1	1	10	1	410	420	410	1	1
	↓	1	1	1	10	1	410	420	410	1	1
t3~t0	↓	1	0	0	11	0	400	440	490	1	1
	↓	0	0	0	11	0	450	440	490	1	1
	↓	0	0	0	11	0	100	440	490	1	1

Note: 1. If the PPGTC/PPGTD is larger than PPGTA, the PPGTA increases to approach PPGTC/PPGTD. When the PPGTC/PPGTD minus PPGTA is less than PPGSA, the PPGTA will add the PPGSA value after PPG being triggered. At this point the PPGTA value will be greater than PPGTC/PPGTD, the PPGTA is equal to PPGTC/PPGTD at the next PPGTA trigger.

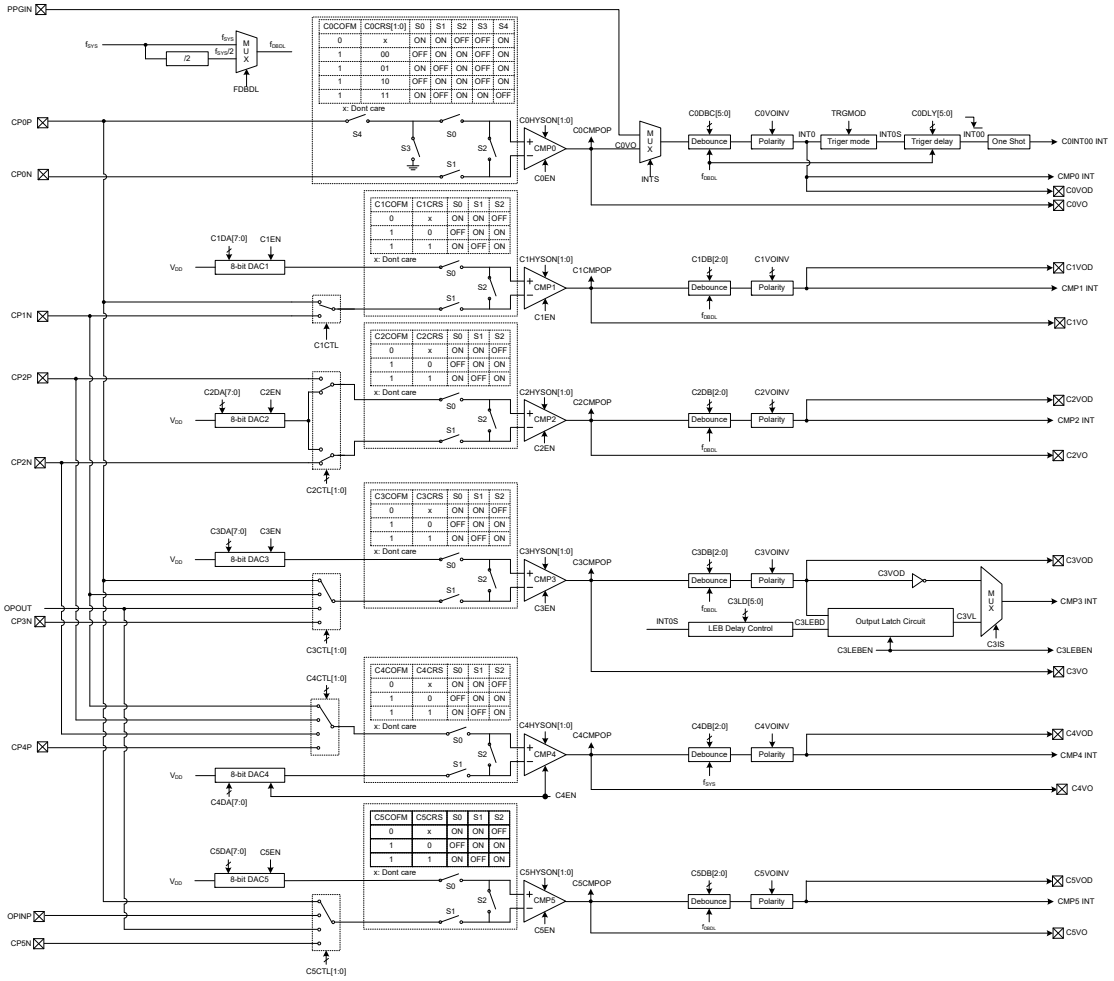
2. If the PPGTC/PPGTD is less than PPGTA, the PPGTA decreases to approach PPGTC/PPGTD. When the PPGTA minus the PPGTC/PPGTD is less than PPGSA, the PPGTA decreases the PPGSA value after PPG being triggered, at this point the PPGTC/PPGTD is larger than the PPGTA, the PPGTA is equal to PPGTC/PPGTD after the next PPG trigger.

3. If PPGTA+PPGSA is larger than 511 or PPGTA+PPGSA is less than 0, the extreme value 511 or 0 will be written to PPGTA directly.


Zero crossing by Software
Zero crossing by Software
Zero crossing by Software
CMP4DA Code
t0
t1
t2
t3
t0
t1
t2
t3
t0
C4VOD
PPGTON
PPGADJF
PPGTMMD[1:0]
11B
00B
01B
10B
11B
00B
01B
10B
11B
PPGTIMER Counter
PPGTMR1
PPGTMR2
PPGTMR3
PPGTMR1
PPGTMR2
PPGTMR3
PPGTC
PPGTC (New)
PPGTD (New)
PPGTA
PPGTD
PPGTA (New)
PPGACF
Cleared by software
Cleared by hardware
PPGADF
Cleared by software
Cleared by hardware

Comparators

The device has six integrated comparators, known as CMP0~CMP5. Five of the comparators, CMP1~CMP5, which have a D/A converter, are used for over voltage protection functions, which provide protection mechanisms or generate output signals for different applications, such as over voltage detection, surge voltage detection, over current detection (valley detection) and zero crossing detection. The remaining comparator, CMP0, is used for synchronous signal detection.



Note: 1. The Comparator 0~3, 5 interrupts are triggered by the C0VOD~C3VOD, C5VOD falling edge while the Comparator 4 interrupt is triggered by the C4VOD rising edge.

2. The comparator external pins are pin-shared with other functions, therefore before using the comparator function, ensure that the pin-shared function registers have been set properly to enable the comparator pin function.

3. The OPOUT is sourced from the Operational Amplifier output.

4. The OPINP is the Operational Amplifier non-inverting input pin.

5. The C0INT00INT is internally connected to the Programmable Pulse Generator and Timer/Event Counter 1/3. The C0VOD is internally connected to the Timer/Event Counter 0/3. The C4VOD is internally connected to the A/D Converter. The C1VOD, C2VOD, C3VOD, C5VOD, C4VOD and C3LEBEN are internally connected to the Programmable Pulse Generator.

Comparator Block Diagram

Comparator Registers

The overall operation of the internal comparators is controlled using a series of registers.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CMP0C0	C0CMPOP	C0COFM	C0COF5	C0COF4	C0COF3	C0COF2	C0COF1	C0COF0
CMP0C1	C0EN	C0HYSON1	C0HYSON0	TRGMOD	C0VOINV	INTS	C0CRS1	C0CRS0
CMP0DB	FDBDL	—	C0DBC5	C0DBC4	C0DBC3	C0DBC2	C0DBC1	C0DBC0
CMP0DLY	—	—	C0DLY5	C0DLY4	C0DLY3	C0DLY2	C0DLY1	C0DLY0
CMPnC0 (n=1~5)	CnCMPOP	CnCOFM	CnCRS	CnCOF4	CnCOF3	CnCOF2	CnCOF1	CnCOF0
CMPnC1 (n=1~5)	CnEN	CnHYSON1	CnHYSON0	CnVOD	CnVOINV	CnDB2	CnDB1	CnDB0
CnDA (n=1~5)	D7	D6	D5	D4	D3	D2	D1	D0
C3LEBC	C3LEBEN	C3IS	C3LD5	C3LD4	C3LD3	C3LD2	C3LD1	C3LD0
CMPCTL0	C4CTL1	C4CTL0	C3CTL1	C3CTL0	C2CTL1	C2CTL0	—	C1CTL
CMPCTL1	—	—	—	—	—	—	C5CTL1	C5CTL0

Comparator Register List

• CMP0C0 Register

Bit	7	6	5	4	3	2	1	0
Name	C0CMPOP	C0COFM	C0COF5	C0COF4	C0COF3	C0COF2	C0COF1	C0COF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **C0CMPOP**: Comparator 0 digital output
0: Positive input voltage < negative input voltage
1: Positive input voltage > negative input voltage

Bit 6 **C0COFM**: Comparator 0 operating mode selection
0: Comparator mode
1: Input offset voltage calibration mode

Bit 5~0 **C0COF5~C0COF0**: Comparator 0 input voltage offset calibration setting

• CMP0C1 Register

Bit	7	6	5	4	3	2	1	0
Name	C0EN	C0HYSON1	C0HYSON0	TRGMOD	C0VOINV	INTS	C0CRS1	C0CRS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	1	0	0

Bit 7 **C0EN**: Comparator 0 enable control
0: Disable
1: Enable
If this bit is cleared to "0", the Comparator 0 will be switched off and no power will be consumed. If this bit is set to "1", the Comparator 0 will be powered.

Bit 6~5 **C0HYSON1~C0HYSON0**: Comparator 0 hysteresis voltage window control bits
Refer to the Comparator Electrical Characteristics section to obtain the exact value.

Bit 4 **TRGMOD**: Select single or double falling edges of INT0 as the input of trigger delay circuit which produces INT00
0: Single falling edge
1: Double falling edges

Bit 3 **C0VOINV**: Inverting control of the comparator 0 output signal
0: Non-inverted
1: Inverted

Bit 2 **INTS**: INT00 source selection
0: PPGIN
1: C0VO

Bit 1~0 **C0CRS1~C0CRS0**: Comparator 0 offset voltage calibration reference input selection
00: Comparator 0 negative input is selected
01: Comparator 0 positive input is selected
10: Comparator 0 negative input is selected
11: Internal 0V input is selected

• **CMP0DB Register**

Bit	7	6	5	4	3	2	1	0
Name	FDBDL	—	C0DBC5	C0DBC4	C0DBC3	C0DBC2	C0DBC1	C0DBC0
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 **FDBDL**: f_{DBDL} clock source selection
0: f_{SYS}
1: $f_{SYS}/2$

Bit 6 Unimplemented, read as "0"

Bit 5~0 **C0DBC5~C0DBC0**: External interrupt input debounce time selection
000000: Bypass digital debounce circuit
000001: $0\sim1/f_{DBDL}$
000010: $1/f_{DBDL}\sim2/f_{DBDL}$
:
:
101111: $46/f_{DBDL}\sim47/f_{DBDL}$
11xxxx: $47/f_{DBDL}\sim48/f_{DBDL}$

• **CMP0DLY Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	C0DLY5	C0DLY4	C0DLY3	C0DLY2	C0DLY1	C0DLY0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **C0DLY5~C0DLY0**: Delay time selection (C0INT00 signal is generated after this delay function)
000000: Bypass digital debounce circuit
000001: $0\sim1/f_{DBDL}$
000010: $1/f_{DBDL}\sim2/f_{DBDL}$
:
:
101111: $46/f_{DBDL}\sim47/f_{DBDL}$
11xxxx: $47/f_{DBDL}\sim48/f_{DBDL}$

• **CMPnC0 Register (n=1~5)**

Bit	7	6	5	4	3	2	1	0
Name	CnCMPOP	CnCOFM	CnCRS	CnCOF4	CnCOF3	CnCOF2	CnCOF1	CnCOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **CnCMPOP**: Comparator n digital output
0: Positive input voltage < negative input voltage
1: Positive input voltage > negative input voltage

Bit 6 **CnCOFM**: Comparator n operating mode selection
0: Comparator mode
1: Input offset voltage calibration mode

Bit 5 **CnCRS**: Comparator n offset voltage calibration reference input selection
0: Comparator 1 negative input is selected
1: Comparator 1 positive input is selected

Bit 4~0 **CnCOF4~CnCOF0**: Comparator n input voltage offset calibration setting

• **CMP4C1 Register**

Bit	7	6	5	4	3	2	1	0
Name	C4EN	C4HYSON1	C4HYSON0	C4VOD	C4VOINV	C4DB2	C4DB1	C4DB0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **C4EN**: Comparator 4 enable control
0: Disable
1: Enable
If this bit is cleared to "0", the Comparator 4 and D/A Converter n will be switched off and no power will be consumed. If this bit is set to "1", the Comparator 4 and D/A Converter n will be powered.

Bit 6~5 **C4HYSON1~C4HYSON0**: Comparator 4 hysteresis voltage window control bits
Refer to the Comparator Electrical Characteristics section to obtain the exact value.

Bit 4 **C4VOD**: Comparator 4 output bit after debounce
If CnVOINV=0:
0: Positive input voltage < negative input voltage
1: Positive input voltage > negative input voltage
If CnVOINV=1:
0: Positive input voltage > negative input voltage
1: Positive input voltage < negative input voltage
This bit stores the comparator 4 output bit after debounce, the status of which is determined by the voltages on the comparator 4 inputs and by the condition of the CnVOINV bit.

Bit 3 **C4VOINV**: Inverting control of the comparator 4 output signal
0: Non-inverted
1: Inverted

Bit 2~0 **C4DB2~C4DB0**: Comparator 4 debounce time selection ($t_{DEB}=1/f_{SYS}$)
000: Bypass, without debounce
001: (3~4)×t_{DEB}
010: (7~8)×t_{DEB}
011: (15~16)×t_{DEB}
100: (31~32)×t_{DEB}
101: (63~64)×t_{DEB}
110: (127~128)×t_{DEB}
111: (255~256)×t_{DEB}

• **CMPnC1 Register – (n=1~3, 5)**

Bit	7	6	5	4	3	2	1	0
Name	CnEN	CnHYSON1	CnHYSON0	CnVOD	CnVOINV	CnDB2	CnDB1	CnDB0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CnEN**: Comparator n enable control
0: Disable
1: Enable
If this bit is cleared to "0", the Comparator n and D/A Converter n will be switched off and no power will be consumed. If this bit is set to "1", the Comparator n and D/A Converter n will be powered.

Bit 6~5 **CnHYSON1~CnHYSON0**: Comparator n hysteresis voltage window control bits
Refer to the Comparator Electrical Characteristics section to obtain the exact value.

Bit 4 **CnVOD**: Comparator n output bit after debounce
If CnVOINV=0:
0: Positive input voltage < negative input voltage
1: Positive input voltage > negative input voltage
If CnVOINV=1:
0: Positive input voltage > negative input voltage
1: Positive input voltage < negative input voltage
This bit stores the comparator n output bit after debounce, the status of which is determined by the voltages on the comparator n inputs and by the condition of the CnVOINV bit.

Bit 3 **CnVOINV**: Inverting control of the comparator n output signal
0: Non-inverted
1: Inverted

Bit 2~0 **CnDB2~CnDB0**: Comparator n debounce time selection ($t_{DBDL}=1/f_{DBDL}$)
000: Bypass, without debounce
001: (3~4)×t_{DBDL}
010: (7~8)×t_{DBDL} t_{DEB}
011: (15~16)×t_{DBDL}
100: (31~32)×t_{DBDL}
101: (63~64)×t_{DBDL}
110: (127~128)×t_{DBDL}
111: (255~256)×t_{DBDL}

• **CMPCTL0 Register**

Bit	7	6	5	4	3	2	1	0
Name	C4CTL1	C4CTL0	C3CTL1	C3CTL0	C2CTL1	C2CTL0	—	C1CTL
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 7~6 **C4CTL1~C4CTL0**: Comparator 4 positive input selection
00: CP1N
01: CP2P
10: CP2N
11: CP4P

Bit 5~4 **C3CTL1~C3CTL0**: Comparator 3 negative input selection
00: CP0P
01: CP1N
10: OPOUT
11: CP3N

Bit 3~2 **C2CTL1~C2CTL0**: Comparator 2 input selection
00: Select DAC2 as negative input and CP2P as positive input
01: Select CP2N as negative input and DAC2 as positive input
1x: Select CP2N as negative input and CP2P as positive input

Bit 1 Unimplemented, read as "0"

Bit 0 **C1CTL**: Comparator 1 negative input selection
0: CP0P
1: CP1N

• CMPCTL1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	C5CTL1	C5CTL0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **C5CTL1~C5CTL0**: Comparator 5 negative input selection
00: CP0P
01: OPINP
10: OPOUT
11: CP5N

• CnDA Register (n=1~5)

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	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**: Comparator n DAC output voltage control bits
DAC V_{OUT}=(DAC V_{REF}/256)×CnDA[7:0]

• C3LEBC Register

Bit	7	6	5	4	3	2	1	0
Name	C3LEBEN	C3IS	C3LD5	C3LD4	C3LD3	C3LD2	C3LD1	C3LD0
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 **C3LEBEN**: Valley detection enable control
0: Disable
1: Enable

Bit 6 **C3IS**: Comparator 3 interrupt signal selection
0: C3VOD
1: C3VL

Bit 5~0 **C3LD5~C3LD0**: C3LEBEN enable LEB delay time selection
000000: Bypass digital delay circuit
000001: 0~2/f_{SYS}
000010: 2/f_{SYS}~4/f_{SYS}
:
:
101111: 92/f_{SYS}~94/f_{SYS}
11xxxx: 94/f_{SYS}~96/f_{SYS}

Comparator Input Offset Calibration Function (n=0, m=5; n=1~5, m=4)

The comparator n includes an input offset calibration function. The calibrated data is stored in the CnCOF[m:0] bits. The CnCOFM is the calibration mode control bit and the CnCRS or C0CRS[1:0] is used to indicate the input reference voltage source in the calibration mode. The CnEN is used to enable or disable the comparator n.

Comparator Offset Calibration Procedure

Note that the hysteresis voltage should be disabled by setting CnHYSON[1:0]=00 before implementing the comparator n offset calibration. As the comparator n inputs are pin-shared with I/O pins, they should be configured as the comparator n inputs first. For comparator n input offset calibration, the procedure is summarised as follows.

Step 1. Set CnEN=1, CnCOFM=1 and CnCRS=1 or C0CRS[1:0]=11, the comparator n is in the offset calibration mode. To make sure V_{CS} as minimize as possible after calibration, the input reference voltage in the calibration should be the same as the input DC operating voltage during normal mode operation.

Step 2. Set CnCOF[m:0]=000000 or 00000 and then read the CnCMPOP bit after a certain delay.

Step 3. Increase the CnCOF[m:0] value by 1 and then read the CnCMPOP bit after a certain delay.

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

If the CnCMPOP bit state has changed, record the CnCOF[m:0] value as V_{CS1} and then go to Step 4.

Step 4. Set CnCOF[m:0]=111111 or 11111 and then read the CnCMPOP bit after a certain delay.

Step 5. Decrease the CnCOF[m:0] value by 1 and then read the CnCMPOP bit after a certain delay.

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

If the CnCMPOP bit state has changed, record the CnCOF[m:0] value as V_{CS2} and then go to Step 6.

Step 6. Restore the Comparator input offset calibration value V_{CS} into the CnCOF[m:0] bits. The offset Calibration procedure has now completed.

When $V_{CS}=(V_{CS1}+V_{CS2})/2$. If $(V_{CS1}+V_{CS2})/2$ is not an integral, discard the decimal.

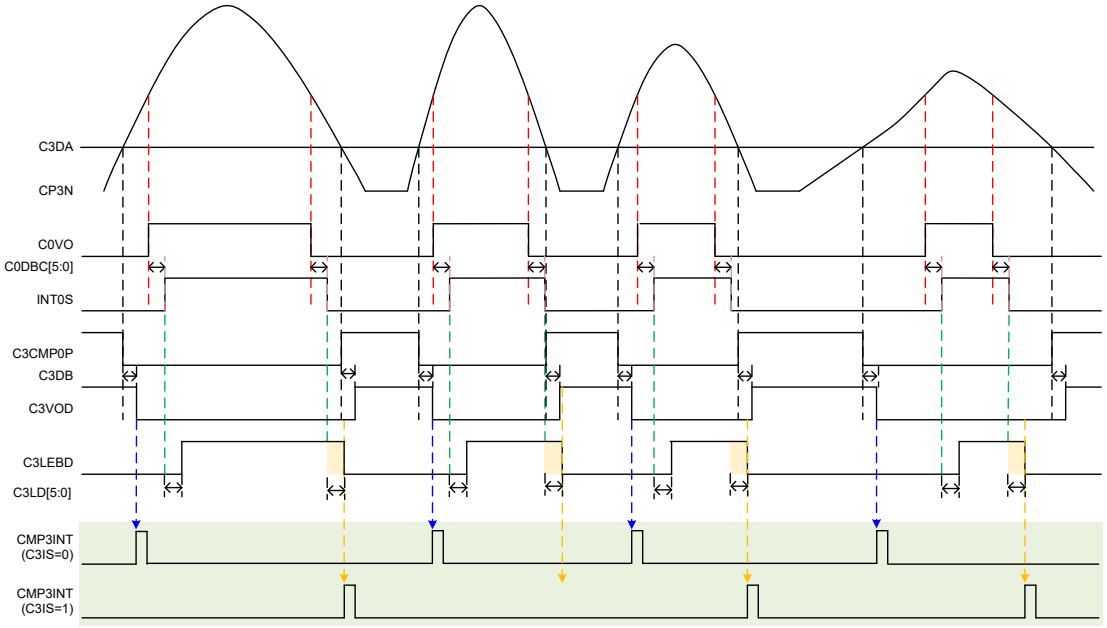
Valley Detection Function

When C3LEBEN=1 and the INT0S has an effective trigger source signal, the C3LEBD signal will be generated after a delay time which is set by C3LD[5:0]. Then the C3LEBD signal is used to detect the C3VOD signal which in turn triggers the interrupt function, as described below:

If C3VOINV=0 and the detected C3VOD signal is “0”, then the CMP3INT will be triggered. If the detected signal is “1”, the CMPINT will not be triggered.

If C3VOINV=1 and the detected C3VOD signal is “1”, then the CMP3INT will be triggered. If the detected signal is “0”, the CMPINT will not be triggered.

The following diagram takes C3VOINV=0 as an example:



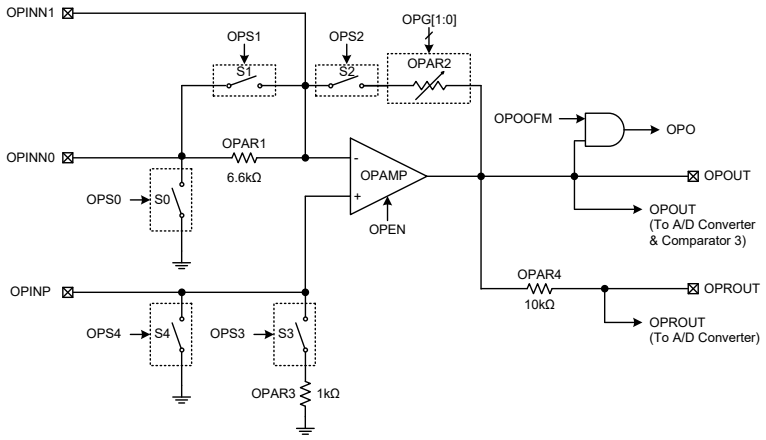
Note: It is recommended to change the C3IS and C3LEBEN bit settings between two PPG triggers (within the PPG active+non-retrigger duration) to avoid unpredictable states of comparator 3 output.

Operational Amplifier

This device includes an operational amplifier for measure applications. An operational amplifier, OPAMP, produces an output potential that is typically hundreds of thousands of times larger than the potential difference between its input terminals. By integrating the OPAMP electronic circuitry into the microcontroller, the need for external components is reduced greatly.

Operational Amplifier Operation

The Operational Amplifier can be used for signal amplification according to specific user requirements. The gain is selectable by using the OPG[1:0] bits. The amplified output can be directly output on the OPOUT and OPROUT pins, and also be internally connected to the A/D converter for the amplified input signal read.



Operational Amplifier Block Diagram

Operational Amplifier Registers

The internal Operational Amplifier normal operation and input offset voltage calibration function are controlled by three registers. The OPS register is used to control the switches on or off. The OPC register is used to control the OPAMP function enable or disable and select the gain. The OPO bit together with the OPOCAL register are used in the offset calibration procedure.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OPC	OPEN	—	—	OPO	—	—	OPG1	OPG0
OPOCAL	OPOOFM	OPORSP	OPOOF5	OPOOF4	OPOOF3	OPOOF2	OPOOF1	OPOOF0
OPS	—	—	—	OPS4	OPS3	OPS2	OPS1	OPS0

Operational Amplifier Register List

• OPC Register

Bit	7	6	5	4	3	2	1	0
Name	OPEN	—	—	OPO	—	—	OPG1	OPG0
R/W	R/W	—	—	R	—	—	R/W	R/W
POR	0	—	—	0	—	—	0	0

Bit 7 **OPEN**: OPAMP function enable control
0: Disable
1: Enable

Bit 6~5 Unimplemented, read as "0"

Bit 4 **OPO**: OPAMP output status (positive logic)
This bit is read only. When the OPOOFM bit is set to 1, the OPO is defined as OPAMP output status, refer to the Input Offset Calibration section.

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **OPG1~OPG0**: R2/R1 ratio selection
00: R2/R1=20
01: R2/R1=30
10: R2/R1=40
11: R2/R1=60
Note that the internal resistors, R1 and R2, should be used when the gain is determined by these bits. This means the OPINN0 pin should be selected and the S2 switch should be on. Otherwise, the gain accuracy will not be guaranteed. (R1=OPAR1; R2=OPAR2)

• OPOCAL Register

Bit	7	6	5	4	3	2	1	0
Name	OPOOFM	OPORSP	OPOOF5	OPOOF4	OPOOF3	OPOOF2	OPOOF1	OPOOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **OPOOFM**: OPAMP operating mode selection
0: Normal operation mode
1: Offset calibration mode

Bit 6 **OPORSP**: OPAMP input offset voltage calibration reference selection
0: Select inverting input as the reference input
1: Select non-inverting input as the reference input

Bit 5~0 **OPOOF5~OPOOF0**: OPAMP input offset voltage calibration control bits
This 6-bit field is used to perform the OPAMP input offset calibration operation and the value for the OPAMP input offset calibration can be restored into this bit field. More detailed information is described in the "Input Offset Calibration" section.

• **OPS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	OPS4	OPS3	OPS2	OPS1	OPS0
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 **OPS4**: OPAMP switch S4 on/off control
0: Off
1: On

Bit 3 **OPS3**: OPAMP switch S3 on/off control
0: Off
1: On

Bit 2 **OPS2**: OPAMP switch S2 on/off control
0: Off
1: On

Bit 1 **OPS1**: OPAMP switch S1 on/off control
0: Off
1: On

Bit 0 **OPS0**: OPAMP switch S0 on/off control
0: Off
1: On

Input Voltage Range

Together with different PGA operating modes, the input voltage on the OPAMP pins can be positive or negative for flexible application. There are two operating mode examples below. In these examples the internal resistors, R1 and R2, are used respectively.

- When the S0 and S2 switches are ON, the S1, S3 and S4 switches are OFF, then the input node is OPINP. For V_{IN}>0, the PGA operates in the non-inverting mode and the output voltage of the PGA is obtained using the formula below:

$$VO_{PGA}=(1+R2/R1)\times V_{IN}$$

- When the S2 and S4 switches are ON, the S0, S1 and S3 switches are OFF, then the input node is OPINN0. For 0>V_{IN}>-0.2V, the PGA operates in the inverting mode, the output voltage of the PGA is obtained using the formula below:

$$VO_{PGA}=-(R2/R1)\times V_{IN}$$

Note that if V_{IN} is negative, it cannot be lower than -0.2V which will result in current leakage.

Input Offset Calibration

This OPAMP includes an input offset calibration function. The calibrated data is stored in the OPOOF[5:0] bits. The OPOOFM bit is the calibration mode control bit and the OPORSP bit is used to indicate that the input reference voltage comes from non-inverting or inverting input in the calibration mode. The OPINP is the OPAMP non-inverting input and the OPINN0 and OPINN1 are the OPAMP inverting inputs. OPOUT and OPROUT are the OPAMP analog voltage output pins. The OPAMP digital output flag is OPO, which is used for OPAMP calibration mode.

Offset Calibration Procedure

Note that as the OPAMP input pins are pin-shared with other functions, they should be configured as OPAMP inputs first by properly configuring the related pin-shared control bits. For operational amplifier input offset calibration, the procedures are summarised as follows.

Step 1. Set OPOOFM=1 and OPORSP=1, the OPAMP is now under offset calibration mode. To make sure V_{OS} as minimal as possible after calibration, the input reference voltage in calibration should be the same as the input DC operating voltage in normal mode operation.

Step 2. Set OPOOF[5:0]=000000 and then read the OPO bit after a certain delay.

Step 3. Increase the OPOOF[5:0] value by 1 and then read the OPO bit after a certain delay.

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

If the OPO bit state has changed, record the OPOOF[5:0] value as V_{OS1} and then go to Step 4.

Step 4. Set OPOOF[5:0]=111111 then read the OPO bit after a certain delay.

Step 5. Decrease the OPOOF[5:0] value by 1 and then read the OPO bit after a certain delay.

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

If the OPO bit state has changed, record the OPOOF[5:0] value as V_{OS2} and then go to Step 6.

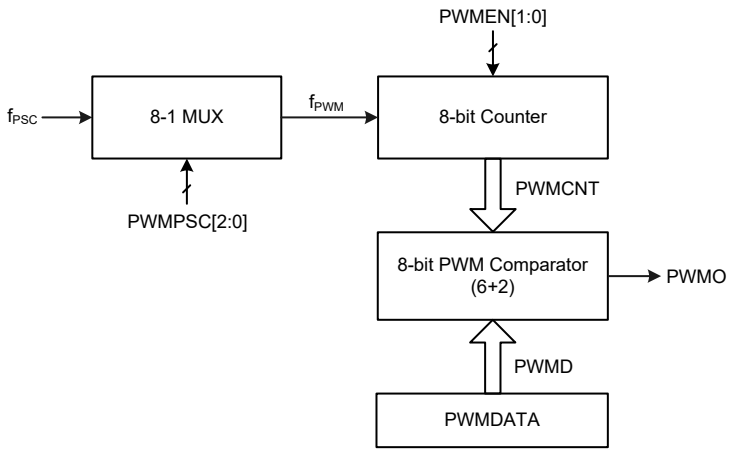
Step 6. Restore the Operational Amplifier input offset calibration value V_{OS} into the OPOOF[5:0] bit field. The offset Calibration procedure is now finished.

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

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

Pulse Width Modulator

The device has one 8-bit pulse width modulation function. Useful for applications such as buzzer control, the PWM function provides an output with a fixed frequency but with a duty cycle that can be varied by setting particular values into the PWMDATA register.



PWM Block Diagram

PWM Register Description

There two registers control the overall operation of the Pulse Width Modulator channel. These are the data register, PWMDATA and a single control register, PWMC.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PWMC	PWMEN1	PWMEN0	—	—	—	PWMPSC2	PWMPSC1	PWMPSC0
PWMDATA	D7	D6	D5	D4	D3	D2	D1	D0

PWM Register List

• **PWMC Register**

Bit	7	6	5	4	3	2	1	0
Name	PWMEN1	PWMEN0	—	—	—	PWMPSC2	PWMPSC1	PWMPSC0
R/W	R/W	R/W	—	—	—	R/W	R/W	R/W
POR	0	0	—	—	—	0	0	0

Bit 7~6 **PWMEN1~PWMEN0**: PWM enable control
00: Disable, output low
01: Disable, output high
1x: Enable
Note that after the PWMEN[1:0] bits are enabled, the first PWM modulation cycle period and duty may not match the expected waveform. The PWM output will be normal after the first PWM cycle.

Bit 5~3 Unimplemented, read as "0"

Bit 2~0 **PWMPSC2~PWMPSC0**: PWM clock, f_{PWM}, prescaler rate selection
000: f_{PSC}
001: $f_{PSC}/2$
010: $f_{PSC}/4$
011: $f_{PSC}/8$
100: $f_{PSC}/16$
101: $f_{PSC}/32$
110: $f_{PSC}/64$
111: $f_{PSC}/128$

• **PWMDATA 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**: PWM output duty cycle PWMD bit 7 ~ bit 0
Note that the duty cycle value PWMD, once being changed, will reflect on the PWMO output signal immediately. Therefore, a sudden PWM duty change will occur in the current PWM cycle, resulting in undesired waveform, which only lasts for a PWM cycle. Starting from the next new PWM cycle, the PWM duty will be in accordance with the new PWMD value.

PWM Operation

The PWMC register and PWMDATA register are assigned to the Pulse Width Modulator channel. The PWM channel has a data register, PWMDATA, the content of which is an 8-bit data, abbreviated as PWMD, representing the overall duty cycle of one modulation cycle of the output waveform. To increase the PWM modulation frequency, each modulation cycle is subdivided into four individual modulation subsections, known as the (6+2) bits mode. The PWM counter clock frequency, f_{PWM}, comes from fPSC or its division. The clock source selection and the enable/disable control for the PWM channel are selected using the PWMC register. Note that when using the PWM, it is only necessary to write the required value into the PWMDATA register and select the clock source and enable/disable control using the PWMC register, the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware.

This method of dividing the original modulation cycle into a further four sub-cycles enables the generation of higher PWM frequencies which allow a wider range of applications to be served. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood. As the PWM clock is f_{PWM}, and as the PWM value is 8-bit wide, the

overall PWM cycle frequency is $f_{PWM}/256$. However, when in the (6+2) mode the PWM modulation frequency will be $f_{PWM}/64$.

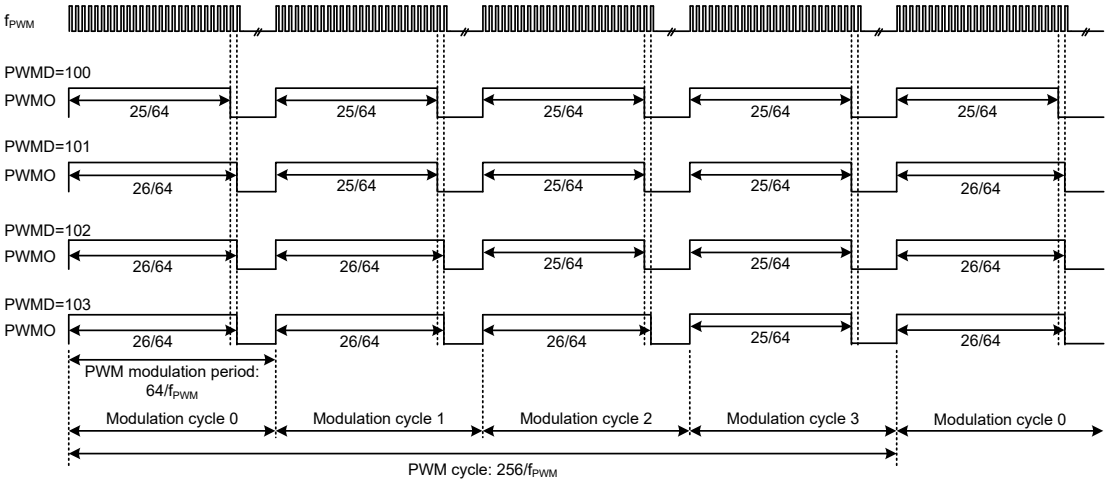
(6+2) Bits PWM Mode Modulation

A (6+2) bits mode PWM cycle is divided into four modulation cycles, which are named as Modulation cycle 0 ~ Modulation cycle 3. Each modulation cycle has 64 PWM input clock periods. In the (6+2) bits PWM mode, the PWMD is divided into two groups. Group 1 is denoted by DC which is the value of PWMD bit 7 ~ bit 2. Group 2 is denoted by AC which is the value of PWMD bit 1 ~ bit 0. The modulation frequency, modulation cycle duty, PWM cycle frequency and PWM cycle duty of the (6+2) bits mode PWM output signals are summarized in the following table.

Modulation Frequency	Modulation Cycle i	Modulation Cycle Duty		PWM Cycle Frequency	PWM Cycle Duty
$f_{PWM}/64$	i=0~3	i<AC	(DC+1)/64	$f_{PWM}/256$	PWMD/256
		i≥AC	DC/64		

(6+2) Bits PWM Mode Summary

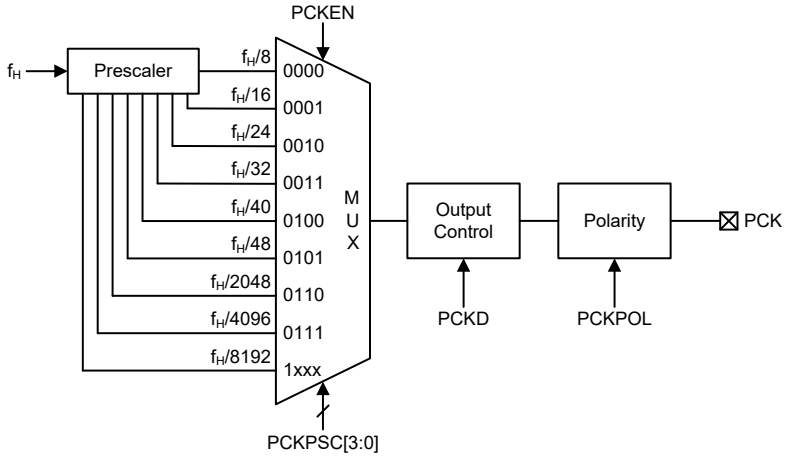
The following diagram illustrates the waveforms associated with the (6+2) bits mode of PWM operation. It is important to note how the single PWM cycle is subdivided into 4 individual modulation cycles, numbered from 0~3 and how the AC value is related to the PWM value. The waveforms of PWM outputs are as shown below.



(6+2) Bits PWM Mode Modulation Waveform

Peripheral Clock Output

The Peripheral Clock Output allows the device to supply external hardware with a clock signal synchronised to the microcontroller clock.



Peripheral Clock Output Block Diagram

Peripheral Clock Output Operation

The peripheral clock output pin PCK is pin-shared with the I/O pin, the pin should be configured as PCK output function by configuring the relevant pin-shared function control bits. The peripheral clock output function is controlled by the PCKC register. After the PCKEN bit has been set, writing a high value to the PCKD bit will enable the PCK output function, writing a zero value will disable the PCK output function and force the output low or high by the PCKPOL bit.

The clock source for the Peripheral Clock Output can originate from the subdivided version of f_H. The division ratio value is determined by the PCKPSC3~PCKPSC0 bits in the PCKC register.

The PCK output truth table is shown below:

PCKEN	PCKD	PCKPOL	PCK Output
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	PCK
1	1	1	PCK

Peripheral Clock Output Register

The peripheral clock output function is controlled by the PCKC register.

- **PCKC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	PCKD	PCKPOL	PCKEN	PCKPSC3	PCKPSC2	PCKPSC1	PCKPSC0
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 **PCKD**: PCK output control
0: Inactive
1: Active
This bit is used to control the PCK output active or inactive. If this bit is cleared to zero, the PCK output status is determined by the PCKPOL bit.

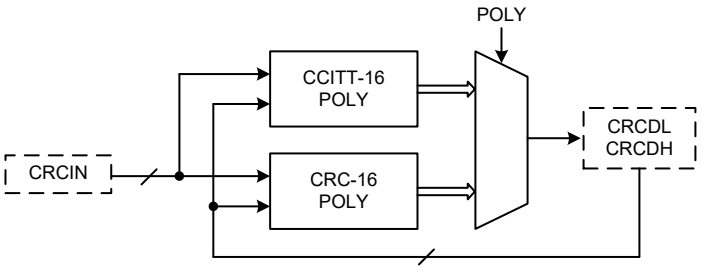
Bit 5 **PCKPOL**: PCK polarity control
0: Non-inverted
1: Inverted
When PCKD=0, if this bit is low, the PCK output is forced to low; if this bit is high, the PCK output us forced to high.

Bit 4 **PCKEN**: PCK function enable control
0: Disable
1: Enable

Bit 3~0 **PCKPSC3~PCKPSC0**: Peripheral clock, f_{PCK}, prescaler selection
0000: $f_H/8$
0001: $f_H/16$
0010: $f_H/24$
0011: $f_H/32$
0100: $f_H/40$
0101: $f_H/48$
0110: $f_H/2048$
0111: $f_H/4096$
1xxx: $f_H/8192$

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
CRCCR	—	—	—	—	—	—	—	POLY
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0

Register Name	Bit							
	7	6	5	4	3	2	1	0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D7	D6	D5	D4	D3	D2	D1	D0

CRC Register List

- **CRCCR Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **POLY**: 16-bit CRC generating polynomial selection
0: CRC-CCITT: $X^{16}+X^{12}+X^5+1$
1: CRC-16: $X^{16}+X^{15}+X^2+1$

- **CRCIN Register**

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

Bit 7~0 **D7~D0**: CRC input data register

- **CRCDL Register**

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

Bit 7~0 **D7~D0**: 16-bit CRC checksum low byte data register

- **CRCDH Register**

Bit	7	6	5	4	3	2	1	0
Name	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 high byte data register

CRC Operation

The CRC generator provides the 16-bit CRC result calculation based on the CRC16 and CCITT CRC16 polynomials. In this CRC generator, there are only these two polynomials available for the numeric values calculation. It cannot support the 16-bit CRC calculations based on any other polynomials.

The following two expressions can be used for the CRC generating polynomial which is determined using the POLY bit in the CRC control register, CRCCR. The CRC calculation result is called as the CRC checksum, CRCSUM, and stored in the CRC checksum register pair, CRCDH and CRCDL.

- CRC-CCITT: $X^{16}+X^{12}+X^5+1$.
- CRC-16: $X^{16}+X^{15}+X^2+1$.

CRC Computation

Each write operation to the CRCIN register creates a combination of the previous CRC value stored in the CRCDH and CRCDL registers and the new data input. The CRC unit calculates the CRC data register value is based on byte by byte. It will take one MCU instruction cycle to calculate the CRC checksum.

CRC Calculation Procedures:

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Execute an "Exclusive OR" operation with the 8-bit input data byte and the 16-bit CRCSUM high byte. The result is called the temporary CRCSUM.
3. Shift the temporary CRCSUM value left by one bit and move a "0" into the LSB.
4. Check the shifted temporary CRCSUM value after procedure 3.

   If the MSB is 0, then this shifted temporary CRCSUM will be considered as a new temporary CRCSUM.

   Otherwise, execute an "Exclusive OR" operation with the shifted temporary CRCSUM in procedure 3 and a data "8005H". Then the operation result will be regarded as the new temporary CRCSUM.

   Note that the data to be perform an "Exclusive OR" operation is "8005H" for the CRC-16 polynomial while for the CRC-CCITT polynomial the data is "1021H".
5. Repeat the procedure 3 ~ procedure 4 until all bits of the input data byte are completely calculated.
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.

Low Voltage Detector – LVD

The device has a Low Voltage Detector function, also known as LVD. This enabled the device to monitor the power supply voltage, V_{DD}, or the LVDIN pin input voltage, and provide a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of eight fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} voltage or the LVDIN pin input voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

• LVDC Register

Bit	7	6	5	4	3	2	1	0
Name	TLVD1	TLVD0	LVDO	LVDEN	VBGEN	VLVD2	VLVD1	VLVD0
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **TLVD1~TLVD0**: Minimum low voltage width to interrupt time, t_{LVD}, selection
00: (1~2)×t_{LIRC}
01: (3~4)×t_{LIRC}
10: (7~8)×t_{LIRC}
11: (1~2)×t_{LIRC}

Bit 5 **LVDO**: LVD output flag
0: No Low Voltage Detected
1: Low Voltage Detected

Bit 4 **LVDEN**: Low voltage detector enable control
0: Disable
1: Enable

Bit 3 **VBGEN**: Bandgap voltage output enable control
0: Disable
1: Enable

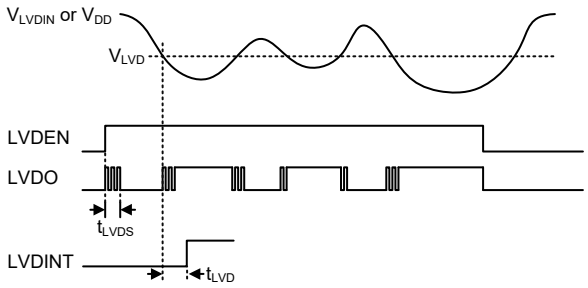
Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VBGEN bit is set high.

Bit 2~0 **VLVD2~VLVD0**: LVD voltage selection
000: $V_{LVDIN} \leq 1.23V$
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

When the VLVD2~VLVD0 bits are set to 000B, the LVD function will be implemented by comparing the LVDIN pin input voltage with the LVD reference voltage of 1.23V. When the VLVD2~VLVD0 bits are set to any other value except 000B, the LVD function will operate by comparing the V_{DD} voltage level with the LVD reference voltage with a specific voltage value which is generated by the internal LVD circuit.

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, or the LVDIN pin input voltage, with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.23V and 4.0V. When the power supply voltage, V_{DD}, or the LVDIN pin input voltage, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. When the device enters the SLEEP mode, the low voltage detector will be automatically disabled even if the LVDEN bit is set high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} or the LVDIN pin input voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



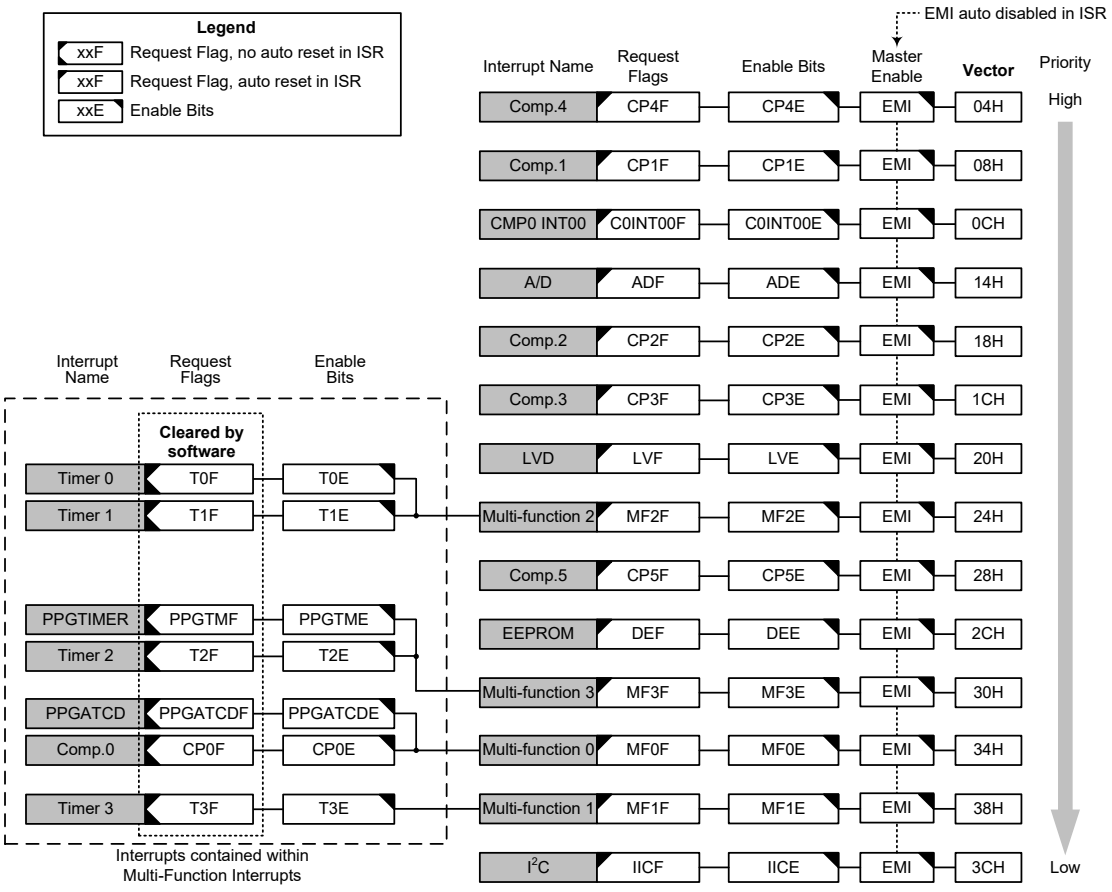
LVD Operation

The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. The actual t_{LVD} value can be selected by the TLVD1~TLVD0 bits. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} or the LVDIN pin input voltage falls below the preset LVD voltage. This will cause the device to wake up from the IDLE Mode, however if the Low Voltage Detector wake-up function is not required then the LVF flag should be first set high before the device enters the IDLE Mode.

Interrupts

Interrupts are an important part of any microcontroller system. When an internal function such as a Timer 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 internal interrupt functions, which are generated by various internal functions such as the Timer/Event Counters, Comparators, LVD, EEPROM 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 registers fall into two categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFIn registers which setup the Multi-function interrupts.

Each register contains a number of enable bits to enable or disable individual registers as well as interrupt flags to indicate the presence of an interrupt request. The naming convention of these follows a specific pattern. First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an "E" for enable/disable bit or "F" for request flag.

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
Comparator	CPnE	CPnF	n=0~5
CMP0 INT00	C0INT00E	C0INT00F	—
PPGTIMER	PPGTME	PPGTMF	—
PPGATCD	PPGATCDE	PPGATCDF	—
A/D Converter	ADE	ADF	—
EEPROM erase or write operation	DEE	DEF	—
LVD	LVE	LVF	—
Multi-function	MFnE	MFnF	n=0~3
Timer/Event Counter	TnE	TnF	n=0~3
I^2C	IICE	IICF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTC0	—	C0INT00F	CP1F	CP4F	C0INT00E	CP1E	CP4E	EMI
INTC1	CP3F	CP2F	ADF	D4	CP3E	CP2E	ADE	D0
INTC2	DEF	CP5F	MF2F	LVF	DEE	CP5E	MF2E	LVE
INTC3	IICF	MF1F	MF0F	MF3F	IICE	MF1E	MF0E	MF3E
MFI0	—	—	CP0F	PPGATCDF	—	—	CP0E	PPGATCDE
MFI1	—	—	T3F	D4	—	—	T3E	D0
MFI2	—	—	T1F	T0F	—	—	T1E	T0E
MFI3	—	—	T2F	PPGTMF	—	—	T2E	PPGTME

Interrupt Register List

- **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	C0INT00F	CP1F	CP4F	C0INT00E	CP1E	CP4E	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 **C0INT00F**: CMP0 INT00 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **CP1F**: Comparator 1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CP4F**: Comparator 4 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **C0INT00E**: CMP0 INT00 interrupt control
0: Disable
1: Enable

Bit 2 **CP1E**: Comparator 1 interrupt control
0: Disable
1: Enable

Bit 1 **CP4E**: Comparator 4 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	CP3F	CP2F	ADF	D4	CP3E	CP2E	ADE	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 **CP3F**: Comparator 3 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **CP2F**: Comparator 2 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **ADF**: A/D converter interrupt request flag
0: No request
1: Interrupt request

Bit 4 **D4**: Reserved, must be fixed at "0"

Bit 3 **CP3E**: Comparator 3 interrupt control
0: Disable
1: Enable

Bit 2 **CP2E**: Comparator 2 interrupt control
0: Disable
1: Enable

Bit 1 **ADE**: A/D converter interrupt control
0: Disable
1: Enable

Bit 0 **D0**: Reserved, must be fixed at "0"

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	DEF	CP5F	MF2F	LVF	DEE	CP5E	MF2E	LVE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **DEF**: Data EEPROM interrupt request flag
0: No request
1: Interrupt request

Bit 6 **CP5F**: Comparator 5 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **MF2F**: Multi-function 2 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **LVF**: LVD interrupt request flag
0: No request
1: Interrupt request

Bit 3 **DEE**: Data EEPROM interrupt control
0: Disable
1: Enable

Bit 2 **CP5E**: Comparator 5 interrupt control
0: Disable
1: Enable

Bit 1 **MF2E**: Multi-function 2 interrupt control
0: Disable
1: Enable

Bit 0 **LVE**: LVD interrupt control
0: Disable
1: Enable

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	IICF	MF1F	MF0F	MF3F	IICE	MF1E	MF0E	MF3E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **IICF**: I^2C interrupt request flag
0: No request
1: Interrupt request

Bit 6 **MF1F**: Multi-function 1 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **MF0F**: Multi-function 0 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **MF3F**: Multi-function 3 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **IICE**: I^2C interrupt control
0: Disable
1: Enable

Bit 2 **MF1E**: Multi-function 1 interrupt control
0: Disable
1: Enable

Bit 1 **MF0E**: Multi-function 0 interrupt control
0: Disable
1: Enable

Bit 0 **MF3E**: Multi-function 3 interrupt control
0: Disable
1: Enable

• **MFI0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CP0F	PPGATCDF	—	—	CP0E	PPGATCDE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **CP0F**: Comparator 0 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 **PPGATCDF**: PPGATCD interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **CP0E**: Comparator 0 interrupt control
0: Disable
1: Enable

Bit 0 **PPGATCDE**: PPGATCD interrupt control
0: Disable
1: Enable

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	T3F	D4	—	—	T3E	D0
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **T3F**: Timer/Event Counter 3 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 **D4**: Reserved, must be fixed at "0"

Bit 3~2 Unimplemented, read as "0"

Bit 1 **T3E**: Timer/Event Counter 3 interrupt control
0: Disable
1: Enable

Bit 0 **D0**: Reserved, must be fixed at "0"

• **MFI2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	T1F	T0F	—	—	T1E	T0E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **T3F**: Timer/Event Counter 1 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 **T0F**: Timer/Event Counter 0 interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **T1E**: Timer/Event Counter 1 interrupt control
0: Disable
1: Enable

Bit 0 **T0E**: Timer/Event Counter 0 interrupt control
0: Disable
1: Enable

• **MFI3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	T2F	PPGTMF	—	—	T2E	PPGTME
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **T2F**: Timer/Event Counter 2 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 **PPGTMF**: PPGTIMERinterrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **T2E**: Timer/Event Counter 2 interrupt control
0: Disable
1: Enable

Bit 0 **PPGTME**: PPGTIMER interrupt control
0: Disable
1: Enable

Interrupt Operation

When the conditions for an interrupt event occur, such as a Timer/Event Counter n overflow or A/D conversion completion etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a "JMP" which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a "RETI", which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.

Comparator Interrupts

The device has six comparator interrupts, controlled by the internal Comparator 0~5. The Comparator 0 interrupt is contained within the Multi-function interrupt and the Comparator 1~5 interrupts are individual interrupts.

A comparator 0 interrupt request will take place when the comparator 0 interrupt request flag, CP0F, is set, which occurs when the comparator 0 output bit changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, comparator 0 interrupt enable bit, CP0E, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and the comparator 0 inputs generate a comparator output transition, a subroutine call to the Multi-function interrupt vector, will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. Also the corresponding Multi-function interrupt request flag will be automatically reset. As the CP0F flag will not be automatically cleared, it has to be cleared by the application program.

A comparator 1~5 interrupt request will take place when the comparator 1~5 interrupt request flags, CP1F~CP5F, are set, a situation that will occur when the comparator 1~5 output bits change state. To allow the program to branch to their respective interrupt vector address, the global interrupt enable bit, EMI, and comparator 1~5 interrupt enable bits, CP1E~CP5E, must first be set. When

the interrupt is enabled, the stack is not full and the comparator 1~5 inputs generate a comparator output transition, a subroutine call to the comparator 1~5 interrupt vectors, will take place. When the interrupt is serviced, the comparator 1~5 interrupt request flags, CP1F~CP5F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

CMP0 INT00 Interrupt

A CMP0 INT00 interrupt request will take place when the CMP0 INT00 interrupt request flag, C0INT00F, is set, a situation that will occur when a C0INT00INT signal falling edge is generated. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and CMP0 INT00 interrupt enable bit, C0INT00E, must first be set. When the interrupt is enabled, the stack is not full and the C0INT00INT signal falling edge is produced, a subroutine call to the CMP0 INT00 interrupt vector, will take place. When the CMP0 INT00 interrupt is serviced, the CMP0 INT00 interrupt flag, C0INT00F, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Timer/Event Counter Interrupts

There are four Timer/Event Counter interrupts, the Timer/Event Counter 0~3 interrupt is contained within the Multi-function interrupt.

A Timer/Event Counter 0~3 interrupt request will take place when the Timer/Event Counter 0~3 interrupt request flags, T0F~T3F, are set, which occurs when the Timer/Event Counter 0~3 overflows. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Timer/Event Counter 0~3 interrupt enable bits, T0E~T3E, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and the Timer/Event Counter 0~3 overflow occurs, a subroutine call to the Multi-function interrupt vector, will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. Also the corresponding Multi-function interrupt request flag will be automatically reset. As the T0F~T3F flags will not be automatically cleared, it has to be cleared by the application program.

A/D Converter Interrupt

The A/D Converter interrupt is controlled by the termination of an A/D conversion process. An A/D Converter interrupt request will take place when the A/D Converter interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter interrupt vector, will take place. When the interrupt is serviced, the A/D Converter interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

LVD Interrupt

A LVD interrupt request will take place when the LVD interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage or a low LVDIN input voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI and LVD interrupt enable bit, LVE, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the LVD interrupt vector, will take place. When the LVD interrupt is serviced, the LVD interrupt request flag, LVF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

EEPROM Interrupt

An EEPROM interrupt request will take place when the EEPROM interrupt request flag, DEF, is set, which occurs when an EEPROM erase or write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM interrupt enable bit, DEE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM erase or write cycle ends, a subroutine call to the EEPROM interrupt vector will take place. When the EEPROM interrupt is serviced, the EEPROM interrupt request flag, DEF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

I²C Interrupt

An I²C interrupt request will take place when the I²C interrupt request flag, IICF, is set, which occurs when a byte of data has been received or transmitted by the I²C interface, or an I²C slave address match occurs, or an I²C bus time-out occurs. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the I²C interrupt enable bit, IICE, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the I²C interrupt vector, will take place. When the interrupt is serviced, the I²C interrupt flag, IICF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Multi-function Interrupts

The device has four 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 PPGATCD interrupt, Comparator 0 interrupt, PPGTIMER interrupt and Timer/Event Counter 0~3 interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags MFnF are set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. When the Multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function interrupt 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.

PPGTIMER Interrupt

The PPGTIMER interrupt is contained within the Multi-function interrupt. A PPGTIMER interrupt request will take place when the PPGTIMER interrupt request flag, PPGTMF, is set, a situation that will occur when the PPGTIMER overflows. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and PPGTIMER interrupt enable bit, PPGTME, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and the PPGTIMER overflow occurs, a subroutine call to the Multi-function interrupt vector, will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. Also the corresponding Multi-function interrupt request flag will be automatically reset. As the PPGTMF flag will not be automatically cleared, it has to be cleared by the application program.

PPGATCD Interrupt

The PPGATCD interrupt is contained within the Multi-function interrupt. A PPGATCD interrupt request will take place when the PPGATCD interrupt request flag, PPGATCDF, is set, a situation that will occur when the PPGTA approaches PPGTC/PPGTD has completed. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, PPGATCD interrupt enable bit, PPGATCDE, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and the PPGTA approaches PPGTC/PPGTD has completed, a subroutine call to the Multi-function interrupt vector, will take place. When the PPGATCD interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. Also the corresponding Multi-function interrupt request flag will be automatically reset. As the PPGATCDF flag will not be automatically cleared, it has to be cleared by the application program.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as a low voltage condition or an A/D conversion process finishes 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.

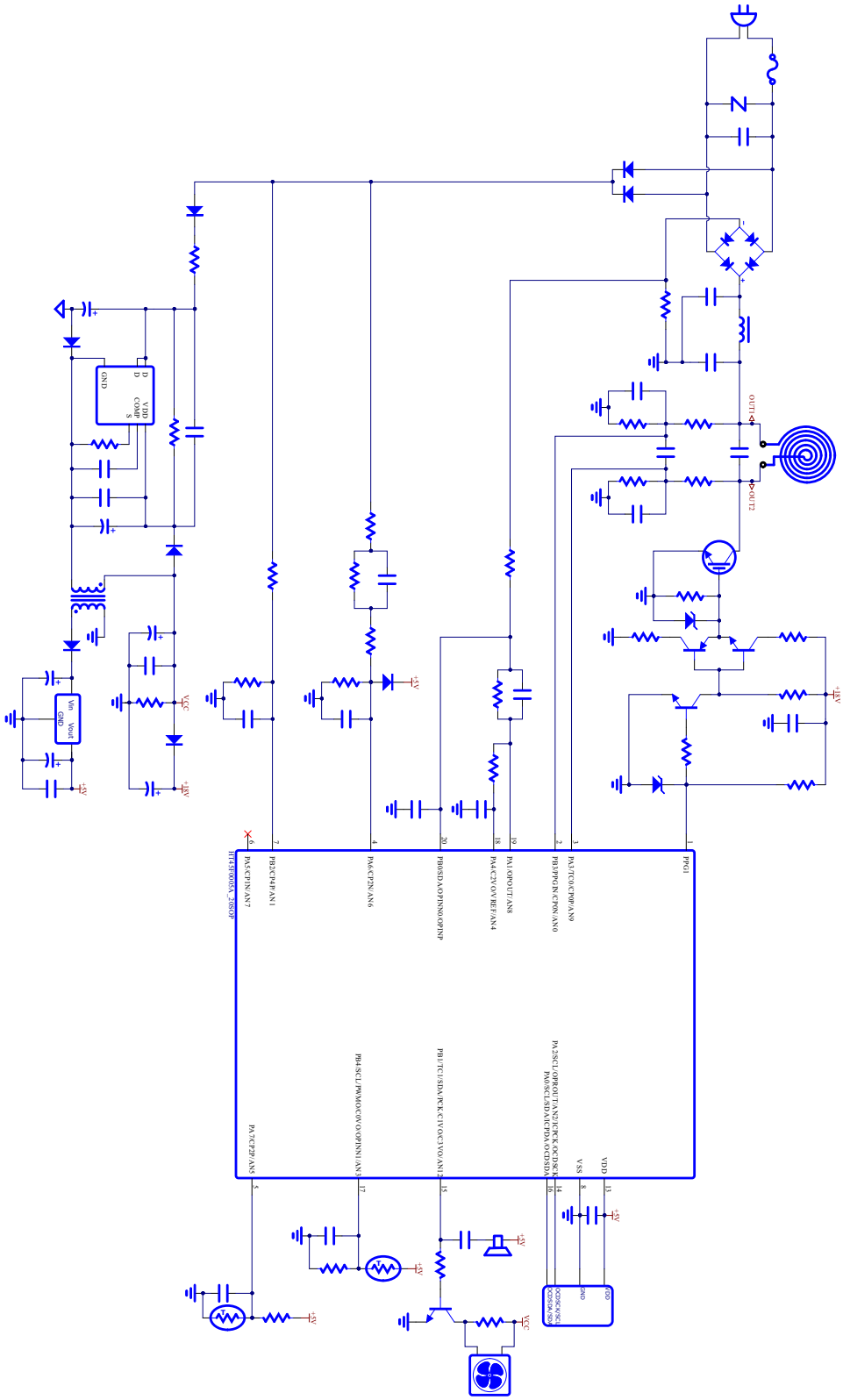
It is recommended that programs do not use the “CALL” instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a CALL subroutine is executed in the interrupt subroutine.

Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode, the wake-up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

Application Circuits



Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of several kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV, SC
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV, SC
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV, SC
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV, SC
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV, SC
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV, SC, CZ
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV, SC, CZ
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV, SC, CZ
SBC A,x	Subtract immediate data from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch Operation			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m]	Skip if Data Memory is not zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read Operation			
TABRD [m]	Read table (specific page) to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
ITABRD [m]	Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory	2Note	None
ITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	2Note	Z, C, AC, OV, SC
LADC A,[m]	Add Data Memory to ACC with Carry	2	Z, C, AC, OV, SC
LADCM A,[m]	Add ACC to Data memory with Carry	2Note	Z, C, AC, OV, SC
LSUB A,[m]	Subtract Data Memory from ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LSBC A,[m]	Subtract Data Memory from ACC with Carry	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2Note	C
Logic Operation			
LAND A,[m]	Logical AND Data Memory to ACC	2	Z
LOR A,[m]	Logical OR Data Memory to ACC	2	Z
LXOR A,[m]	Logical XOR Data Memory to ACC	2	Z
LANDM A,[m]	Logical AND ACC to Data Memory	2Note	Z
LORM A,[m]	Logical OR ACC to Data Memory	2Note	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2Note	Z
LCPL [m]	Complement Data Memory	2Note	Z
LCPLA [m]	Complement Data Memory with result in ACC	2	Z
Increment & Decrement			
LINCA [m]	Increment Data Memory with result in ACC	2	Z
LINC [m]	Increment Data Memory	2Note	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2Note	Z
Rotate			
LRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2Note	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2Note	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2Note	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2Note	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2Note	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2Note	None
LSET [m].i	Set bit of Data Memory	2Note	None

Mnemonic	Description	Cycles	Flag Affected
Branch			
LSZ [m]	Skip if Data Memory is zero	2Note	None
LSZA [m]	Skip if Data Memory is zero with data movement to ACC	2Note	None
LSNZ [m]	Skip if Data Memory is not zero	2Note	None
LSZ [m].i	Skip if bit i of Data Memory is zero	2Note	None
LSNZ [m].i	Skip if bit i of Data Memory is not zero	2Note	None
LSIZ [m]	Skip if increment Data Memory is zero	2Note	None
LSDZ [m]	Skip if decrement Data Memory is zero	2Note	None
LSIZA [m]	Skip if increment Data Memory is zero with result in ACC	2Note	None
LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC	2Note	None
Table Read			
LTABRD [m]	Read table (specific page) to TBLH and Data Memory	3Note	None
LTABRDL [m]	Read table (last page) to TBLH and Data Memory	3Note	None
LITABRD [m]	Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory	3Note	None
LITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	3Note	None
Miscellaneous			
LCLR [m]	Clear Data Memory	2Note	None
LSET [m]	Set Data Memory	2Note	None
LSWAP [m]	Swap nibbles of Data Memory	2Note	None
LSWAPA [m]	Swap nibbles of Data Memory with result in ACC	2	None

Note: 1. For these extended skip instructions, if the result of the comparison involves a skip then three cycles are required, if no skip takes place two cycles is required.

2. Any extended instruction which changes the contents of the PCL register will also require three cycles for execution.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

ADCM A,[m] Add ACC to Data Memory with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

ADD A,[m] Add Data Memory to ACC

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

ADD A,x Add immediate data to ACC

Description The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation ACC ← ACC + x

Affected flag(s) OV, Z, AC, C, SC

ADDM A,[m] Add ACC to Data Memory

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

AND A,[m] Logical AND Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC "AND" x

Affected flag(s) Z

ANDM A,[m] Logical AND ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

Description Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement).
Bits which previously contained a 1 are changed to 0 and vice versa.

Operation [m] ← [m]

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement).
Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC ← [m]

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

DEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

Description This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

OR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" x

Affected flag(s) Z

ORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

RET Return from subroutine

Description The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation Program Counter ← Stack

Affected flag(s) None

RET A,x Return from subroutine and load immediate data to ACC

Description The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

Description The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

RLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

RR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s) C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m] − C

Affected flag(s) OV, Z, AC, C, SC, CZ

SBC A, x Subtract immediate data from ACC with Carry

Description The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC – [m] – C

Affected flag(s) OV, Z, AC, C, SC, CZ

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m] − C

Affected flag(s) OV, Z, AC, C, SC, CZ

SDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

SNZ [m].i Skip if bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m]≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − x

Affected flag(s) OV, Z, AC, C, SC, CZ

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

Description The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

TABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRD [m] Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

XOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "XOR" [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC "XOR" [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "XOR" x

Affected flag(s) Z

Extended Instruction Definition

The extended instructions are used to directly access the data stored in any data memory sections.

LADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

LADCM A,[m] Add ACC to Data Memory with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

LADD A,[m] Add Data Memory to ACC

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

LADDM A,[m] Add ACC to Data Memory

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

LAND A,[m] Logical AND Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

LANDM A,[m] Logical AND ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

LCLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

LCLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

LCPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation [m] ← [m]

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC ← [m]

Affected flag(s) Z

LDAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

LDEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation [m] ← [m] − 1

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] − 1

Affected flag(s) Z

LINC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation [m] ← [m] + 1

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] + 1

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

LMOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

LOR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

LORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

LRL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

LRLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

LRLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

LRLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in theAccumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

LRR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

LRRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s) None

LRRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

LRRCA [m] Rotate Data Memory right through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s) C

LSBC A,[m] Subtract Data Memory from ACC with Carry

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m] − C

Affected flag(s) OV, Z, AC, C, SC, CZ

LSBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m] − C

Affected flag(s) OV, Z, AC, C, SC, CZ

LSDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

LSET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

LSET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

LSIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

LSIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

LSNZ [m].i Skip if bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the content of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m] ≠ 0

Affected flag(s) None

LSUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

LSUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

LSWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

LSWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

LSZ [m] Skip if Data Memory is 0

Description The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

LSZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

LSZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

LTABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LTABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRD [m] Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LXOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "XOR" [m]

Affected flag(s) Z

LXORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC "XOR" [m]

Affected flag(s) Z

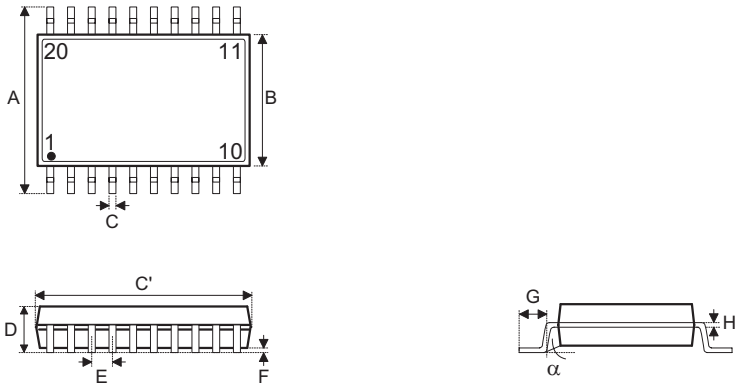
Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- Carton information

20-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.406 BSC	
B		0.295 BSC	
C	0.012	—	0.020
C’		0.504 BSC	
D	—	—	0.104
E		0.050 BSC	
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		10.30 BSC	
B		7.50 BSC	
C	0.31	—	0.51
C’		12.80 BSC	
D	—	—	2.65
E		1.27 BSC	
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°