Sub-1GHz RF Transmitter Flash MCU

BC68F2130/BC68F2140

Revision: V1.10 Date: May 19, 2017

www.holtek.com

Table of Contents

Features

CPU Features

- Operating voltage
  - f_{SYS}=16MHz: 2.0V~3.6V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power saving and wake-up functions to reduce power consumption
- Oscillator types:
  - RF External High Speed Crystal – HXT
  - Internal High Speed RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE, SLEEP and DEEP SLEEP
- Fully integrated internal 16MHz oscillator requires 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: 2K×16 ~ 4K×16
- RAM Data Memory: 256×8
- In Application Programming function – IAP
- Watchdog Timer function
- Up to 14 bidirectional I/O lines
- 2 pin-shared external interrupts
- Multiple Timer Modules for time measurement, input capture, compare match output or PWM output or single pulse output function
  - 1 Compact type 10-bit Timer Module – CTM
  - 1 Periodic type 10-bit Timer Module – PTM
- Dual Time-Base functions for generation of fixed time interrupt signals
- Integrated 1.5V LDO
- Low voltage reset function
- Low voltage detect function
- Flash program memory can be re-programmed up to 100,000 times
- Flash program memory data retention > 10 years
- Package types: 16-pin NSOP-EP, 24-pin SSOP-EP

RF Transmitter Features

- Complete ultra high frequency OOK/FSK transmitter with low transmission phase noise
- Supports 315/433/868/915MHz frequency bands
- Programmable channel setting with < 2kHz resolution
- OOK data rates 0.5Kbps~25Kbps, FSK data rates 0.5Kbps~100Kbps
- Output power up to +13dBm (software controlled output power: 0, +10dBm, +13dBm)
- RF external crystal: 12MHz/12.8MHz/16MHz/19.2MHz

General Description

These Sub-1GHz RF Transmitter MCUs provide a combination of a fully featured MCU and an RF transmitter function, providing them with superior flexibility for use in a wide range of wireless I/O control applications such as industrial control, consumer products, subsystem controllers, etc.

Offering users the convenience of Flash Memory multi-programming features, the devices also include a wide range of functions and features. Other memory includes an area of RAM Data Memory for storage of non-volatile data such as serial numbers, calibration data etc. By using Holtek's In Application Programming technology, users have a convenient means to directly store their measured data in the Flash Program Memory as well as having the ability to easily update their application programs.

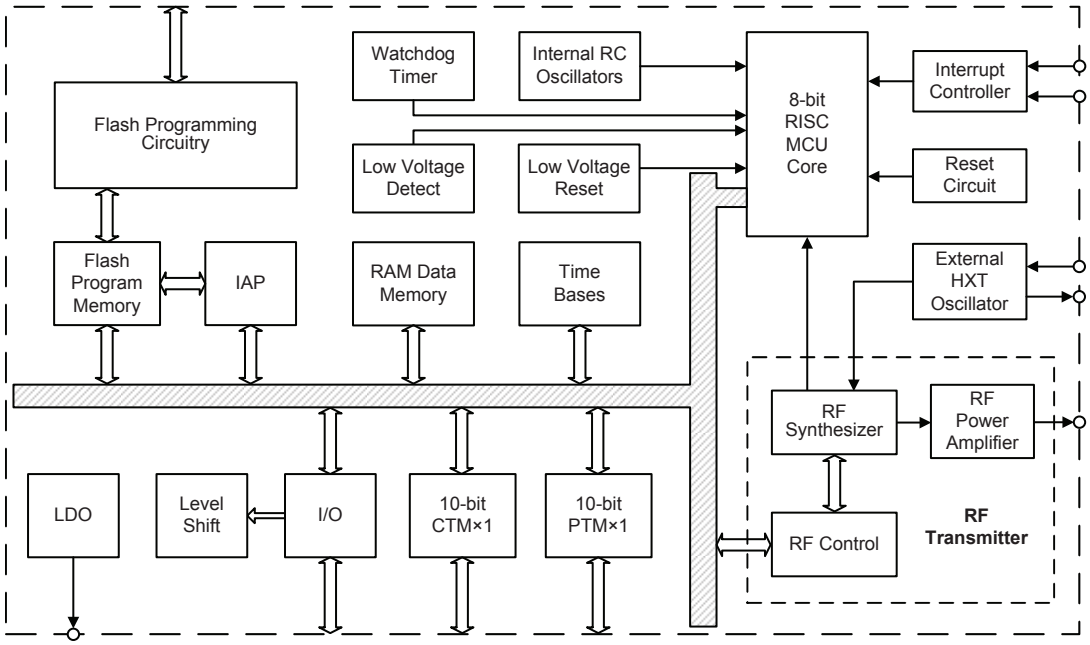
Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

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

The integrated RF transmitter can operate at the 315MHz, 433MHz, 868MHz and 915MHz frequency bands. The additional of a crystal and a limited number of external components is all that is required to create a complete and versatile RF transmitter system. The devices include an internal power amplifier and are capable of delivering +13dBm (Max.) into a 50W load. Such a power level enables a small form factor transmitter to operate near the maximum transmission regulation limits. The devices can operate with OOK – On-Off Keying and FSK – Frequency Shift Keying receiver types. The FSK data rate is up to 100Kbps, allowing the devices to support more complicated control protocols.

The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the devices will be highly capable of providing cost effective MCU Sub-1GHz RF transmitter solutions for remote wireless applications.

Block Diagram



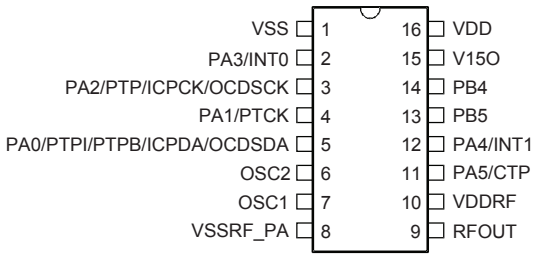
Selection Table

Most features are common to these devices, the main features distinguishing them are Flash Memory capacity, I/O count and the package type. The following table summarises the main features of each device.

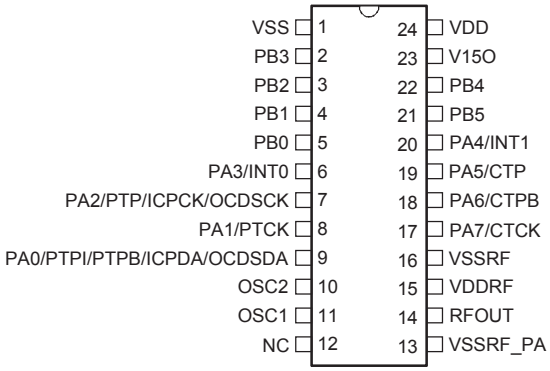
Part No.	V_{DD}	ROM	RAM	I/O	Ext. Int.	LDO
BC68F2130	2.0V~3.6V	2K×16	256×8	8	2	√
BC68F2140	2.0V~3.6V	4K×16	256×8	14	2	√

Part No.	Timer Module	Integrated RF		Time Base	Stack	Package
		Band	Type			
BC68F2130	10-bit CTM×1 10-bit PTM×1	315~915MHz	OOK/FSK TX	2	8	16NSOP-EP
BC68F2140	10-bit CTM×1 10-bit PTM×1	315~915MHz	OOK/FSK TX	2	8	24SSOP-EP

Pin Assignment



BC68F2130
16 NSOP-EP-A



BC68F2140
24 SSOP-EP-A

Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by corresponding software control bits.
2. The OCDSDA and OCDSCK pins are used as the OCDS dedicated pins.

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.

BC68F2130

Pin Name	Function	OPT	I/T	O/T	Description
PA0/PTPI/PTPB/ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTPI	PAS0	ST	—	PTM capture input
	PTPB	PAS0	—	CMOS	PTM inverted output
	ICPDA	—	ST	CMOS	ICP Address/Data
	OCDSDA	—	ST	CMOS	OCDS data/address
PA1/PTCK	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTCK	PAS0	ST	—	PTM clock input

Pin Name	Function	OPT	I/T	O/T	Description
PA2/PTP/ ICPCK/ OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP	PAS0	—	CMOS	PTM output
	ICPCK	—	ST	—	ICP clock
	OCDSCK	—	ST	—	OCDS clock
PA3/INT0	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT0	PAS0	ST	—	External interrupt 0 input
PA4/INT1	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT1	PAS1	ST	—	External interrupt 1 input
PA5/CTP	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP	PAS1	—	CMOS	CTM output
PB4~PB5	PBn	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
OSC1	OSC1	—	HXT	—	RF oscillator input
OSC2	OSC2	—	—	HXT	RF oscillator output
RFOUT	RFOUT	—	—	AN	RF power amplifier output
VDD	VDD	—	PWR	—	3.3V positive power supply
V15O	V15O	—	PWR	—	Internal 1.5V LDO output
VSS	VSS	—	PWR	—	Digital negative power supply
VDDRF	VDDRF	—	PWR	—	RF positive power supply
VSSRF_PA	VSSRF_PA	—	PWR	—	RF power amplifier negative power supply

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

BC68F2140

Pin Name	Function	OPT	I/T	O/T	Description
PA0/PTPI/PTPB/ ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTPI	PAS0	ST	—	PTM capture input
	PTPB	PAS0	—	CMOS	PTM inverted output
	ICPDA	—	ST	CMOS	ICP Address/Data
	OCDSDA	—	ST	CMOS	OCDS data/address
PA1/PTCK	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTCK	PAS0	ST	—	PTM clock input

Pin Name	Function	OPT	I/T	O/T	Description
PA2/PTP/ICPCK/ OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP	PAS0	—	CMOS	PTM output
	ICPCK	—	ST	—	ICP clock
	OCDSCK	—	ST	—	OCDS clock
PA3/INT0	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT0	PAS0	ST	—	External interrupt 0 input
PA4/INT1	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT1	PAS1	ST	—	External interrupt 1 input
PA5/CTP	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP	PAS1	—	CMOS	CTM output
PA6/CTPB	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTPB	PAS1	—	CMOS	CTM inverted output
PA7/CTCK	PA7	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK	—	ST	—	CTM clock input
PB0~PB5	PBn	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
OSC1	OSC1	—	HXT	—	RF oscillator input
OSC2	OSC2	—	—	HXT	RF oscillator output
RFOUT	RFOUT	—	—	AN	RF power amplifier output
VDD	VDD	—	PWR	—	3.3V positive power supply
V15O	V15O	—	PWR	—	Internal 1.5V LDO output
VSS	VSS	—	PWR	—	Digital negative power supply
VDDRF	VDDRF	—	PWR	—	RF positive power supply
VSSRF	VSSRF	—	PWR	—	RF negative power supply
VSSRF_PA	VSSRF_PA	—	PWR	—	RF power amplifier negative power supply
NC	NC	—	—	—	Not connected

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

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}−0.3V to V_{SS}+3.6V
Input Voltage ..... V_{SS}−0.3V to V_{DD}+0.3V
Storage Temperature ..... -50°C to 125°C
Operating Temperature ..... -40°C to 85°C
I_{OH} Total ..... -80mA
I_{OL} Total 80mA
Total Power Dissipation 500mW
ESD HBM ..... ±2kV
ESD MM ..... ±400V

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

The devices are ESD sensitive. HBM (Human Body Mode) is based on MIL-STD-883H Method 3015.8. MM (Machine Mode) is based on JEDEC EIA/JESD22-A115.

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	MCU Operating Voltage – HXT (RF External Crystal)	—	f_{SYS} = f_{HXT} = 12/12.8/16/19.2MHz	2.0	—	3.6	V
	MCU Operating Voltage – HIRC	—	f_{SYS} = f_{HIRC} = 16MHz	2.0	—	3.6	V
	MCU Operating Voltage – LIRC	—	f_{SYS} = f_{LIRC} = 32kHz	2.0	—	3.6	V
V_{15O}	LDO Operating Voltage	2.0V~3.6V	—	-10%	1.5	+10%	V
I_{DD}	Operating Current – LIRC (LCMD=1)	3V	No load, all peripherals off, f_{SYS} = f_{LIRC} = 32kHz, RF TX power off	—	30	50	μA
	Operating Current – LIRC (LCMD=0)	3V	No load, all peripherals off, f_{SYS} = f_{LIRC} = 32kHz, RF TX power off	—	42	70	μA
	Operating Current – HIRC	3V	No load, all peripherals off, f_{SYS} = f_{HIRC} = 16MHz, RF TX power off	—	0.76	1.5	mA
	Operating Current – HXT	3V	No load, all peripherals off, f_{SYS} = f_{HXT} = 16MHz, RF TX power off	—	1.1	1.6	mA
I_{STB}	Standby Current (Deep Sleep Mode, LDO Off)	3V	No load, all peripherals off, WDT off	—	0.4	1.0	μA
	Standby Current (DEEP SLEEP Mode, LDO Off)	3V	No load, all peripherals off, WDT on	—	1.2	3	μA
	Standby Current (IDLE1 Mode, LDO On)	3V	No load, all peripherals off, f_{SUB} on, f_{SYS} = f_{HXT} = 16MHz	—	600	900	μA
V_{IL}	Input Low Voltage for I/O Ports	3V	—	0	—	0.9	V
		—	—	0	—	0.3V_{DD}	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IH}	Input High Voltage for I/O Ports	3V	—	2.1	—	3.0	V
		—	—	$0.7V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Ports	3V	$V_{OL}=0.1V_{DD}$	10	20	—	mA
I_{OH}	Source Current for I/O Ports	3V	$V_{OL}=0.9V_{DD}$	-5	-10	—	mA
V_{OL}	Output Low Voltage for I/O Ports	3V	I_{OL}=10mA	—	—	0.3	V
V_{OH}	Output High Voltage for I/O Ports	3V	I_{OH}=-5mA	2.7	—	—	V
R_{PH}	Pull-high Resistance for I/O Ports	3V	—	20	30	60	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	Sysetm Clock – HXT (RF External Crystal)	2.0V~3.6V	$f_{SYS}=f_{HXT}$ = 12/12.8/16/19.2MHz	12	16	19.2	MHz
	MCU System Clock – HIRC	2.0V~3.6V	$f_{SYS}=f_{HIRC}$ = 16MHz	—	16	—	MHz
	System Clock – LIRC	2.0V~3.6V	$f_{SYS}=f_{LIRC}$ = 32kHz	—	32	—	kHz
f_{HIRC}	High Speed Internal RC Oscillator	3V	Ta = 25°C	-2%	16	+2%	MHz
f_{LIRC}	Low Speed Internal RC Oscillator	3V	Ta = 25°C	-10%	32	+10%	kHz
t_{INT}	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs
t_{TCK}	xTM xTCK Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TPI}	PTM PTPI Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{SST}	System Start-up Timer Period (Wake-up from halt condition where f_{SYS} is off)	—	$f_{SYS}=f_{HXT}$ ~ f_{HXT} / 64	128	—	—	t_{HXT}
		—	$f_{SYS}=f_{HIRC}$ ~ f_{HIRC} / 64	16	—	—	t_{HIRC}
		—	$f_{SYS}=f_{LIRC}$	2	—	—	t_{SUB}
	System Start-up Timer Period (Wake-up from halt condition where f_{SYS} is on)	—	$f_{SYS}=f_H$ ~ f_H / 64, $f_H=f_{HXT}$ or f_{HIRC}	2	—	—	t_H
		—	$f_{SYS}=f_{LIRC}$	2	—	—	t_{SUB}
	System Start-up Timer Period (Slow mode ↔ Normal mode or $f_H=f_{HIRC}$ ↔ f_{HXT})	—	f_{HXT} off → on (HXTF = 1)	1024	—	—	t_{HXT}
		—	f_{HIRC} off → on (HIRCF = 1)	16	—	—	t_{HIRC}
	System Start-up Timer Period (WDT Time-out Hardware Reset)	—	—	0	—	—	t_H
t_{RSTD}	System Reset Delay Time (Power-on Reset, LVR Hardware Reset, LVRC/WDTC/RSTC Software Reset)	—	—	25	50	100	ms
	System Reset Delay Time (WDT Hardware Reset)	—	—	8.3	16.7	33.3	ms
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

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 1.9V	-0.1	1.9	+0.1	V
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 1.9V	-0.1	1.9	+0.1	V
		—	LVD enable, voltage select 2.0V	-5%	2.0	+5%	
		—	LVD enable, voltage select 2.2V		2.2		
		—	LVD enable, voltage select 2.4V		2.4		
		—	LVD enable, voltage select 2.8V		2.8		
		—	LVD enable, voltage select 2.9V		2.9		
		—	LVD enable, voltage select 3.0V		3.0		
		—	LVD enable, voltage select 3.3V		3.3		
I_{OP}	Operating Current	3V	LVD enable, LVR enable	—	25	35	μA
t_{LVDS}	LVDO Stable Time	—	For LVR enable, LVD off → on	—	—	15	μs
		—	For LVR disable, LVD off → on	—	—	150	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs

RF Transmitter Electrical Characteristics

Ta=25°C

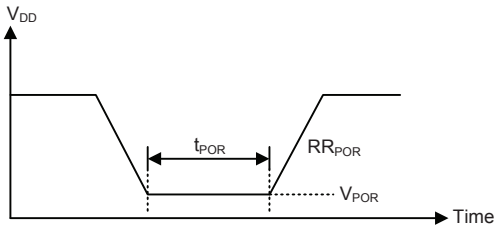
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DDRF}	RF Operating Voltage	—	—	2.0	3.0	3.6	V
I_{STBRF}	RF Power-down Current	—	—	—	—	1	μA
f_{XTAL}	RF Operating XTAL	—	—	12	16	19.2	MHz
f_{XTALV}	RF Operating XTAL Variation	—	—	-20	—	+20	ppm
f_{OP}	RF Operating Frequency	—	—	315	—	915	MHz
f_{FSK}	FSK Frequency Deviation	—	—	10	—	150	kHz
f_{STEP}	RF Frequency Step	—	f_{OP} = 315/433MHz	—	—	50	Hz
		—	f_{OP} = 868/915MHz	—	—	100	
R_{FSK}	Air Data Rate (FSK)	—	—	0.5	—	100	Kbps
R_{OOK}	Air Data Rate (OOK)	—	—	0.5	—	25	Kbps
P_{RF}	Output Power	3.0V	—	0	10	13	dBm
P_{RFV}	Output Power Variation	2.0V~3.3V	P_{RF} = 10dBm (f_{OP} = 433MHz)	—	—	6	dB
		2.5V~3.3V	P_{RF} = 10dBm (f_{OP} = 433MHz)	—	—	3	dB
I_{DDTX}	Transmitter Operating Current	3.0V	P_{RF} = 0dBm (f_{OP} = 315/433MHz)	—	12	—	mA
		3.0V	P_{RF} = 10dBm (f_{OP} = 315/433MHz)	—	18	—	mA
		3.0V	P_{RF} = 13dBm (f_{OP} = 315/433MHz)	—	25	—	mA
		3.0V	P_{RF} = 0dBm (f_{OP} = 868/915MHz)	—	13	—	mA
		3.0V	P_{RF} = 10dBm (f_{OP} = 868/915MHz)	—	22	—	mA
		3.0V	P_{RF} = 13dBm (f_{OP} = 868/915MHz)	—	28	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
PN_{TX}	Transmitter Phase Noise	—	P_{RF} = 10dBm, 100kHz from Carrier	—	—	-80	dBc/Hz
		—	P_{RF} = 10dBm, 1MHz from Carrier	—	—	-100	dBc/Hz
SE_{TX}	Transmitter Spurious Emission (P_{RF} = 10dBm, f_{OP} = 433MHz)	—	f < 1GHz	—	—	-36	dBm
		—	47MHz < f < 74MHz, 87.5MHz < f < 118MHz, 174MHz < f < 230MHz, 470MHz < f < 790MHz,	—	—	-54	dBm
		—	2nd Harmonic	—	—	-30	dBm
		—	3rd Harmonic	—	—	-30	dBm
		—	2nd Harmonic (other frequency)	—	—	-30	dBm
		—	3rd Harmonic (other frequency)	—	—	-30	dBm

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



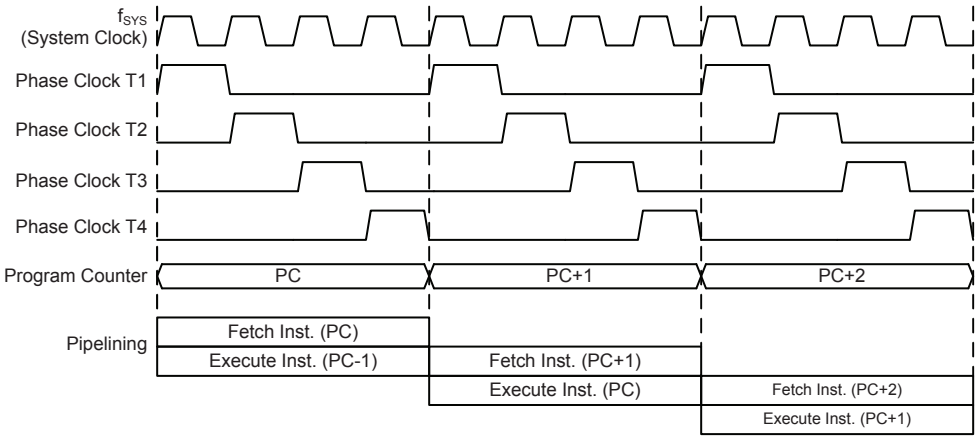
System Architecture

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

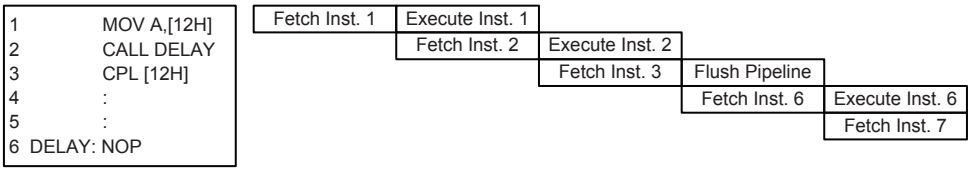
Clocking and Pipelining

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

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



System Clocking and Pipelining



Instruction Fetching

Program Counter

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

Device	Program Counter	
	Program Counter High Byte	PCL Register
BC68F2130	PC10~PC8	PCL7~PCL0
BC68F2140	PC11~PC8	PCL7~PCL0

Program Counter

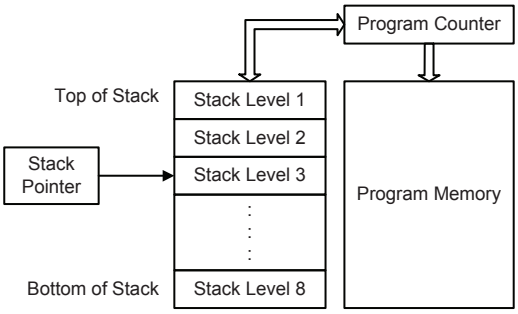
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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.

Stack

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

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

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



Arithmetic and Logic Unit – ALU

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

- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
  LAND, 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

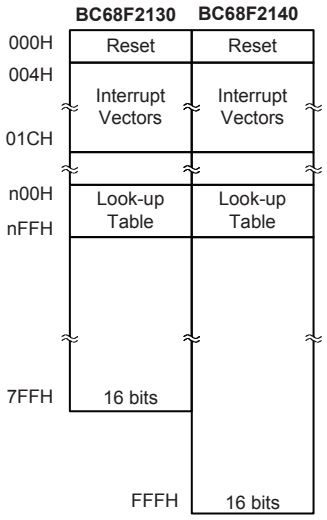
Flash Program Memory

The Program Memory is the location where the user code or program is stored. For the devices the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, the Flash devices offer 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 2K×16 to 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.

Device	Capacity
BC68F2130	2K×16
BC68F2140	4K×16



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

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

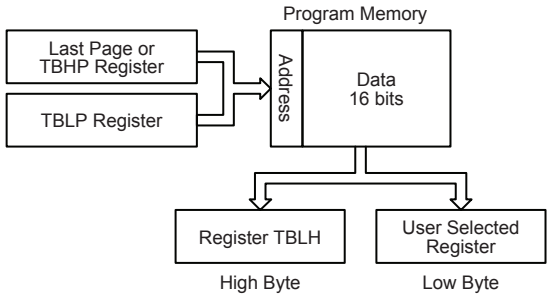


Table Program Example

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 "700H" which refers to the start address of the last page within the 2K words Program Memory of the BC68F2130. 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 "706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the specific page pointed by the TBHP register if the "TABRD [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]" 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

```
ds .section 'data'
tempreg1 db?              ; temporary register #1
tempreg2 db?              ; temporary register #2
code0 .section 'code'
mov a,06h                 ; initialise low table pointer - note that this address is
                          ; referenced
mov tblp,a                ; to the last page or the page that tbhp pointed
mov a,07h                 ; initialise high table pointer
mov tbhp,a                ; it is not necessary to set tbhp if executing tabrdl
:
:
tabrd tempreg1            ; transfers value in table referenced by table pointer
                          ; data at program memory address "706H" 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 "705H" 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 700h                  ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

In Circuit Programming – ICP

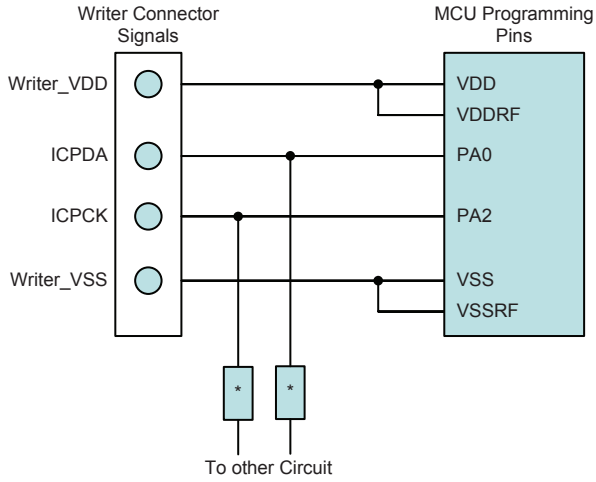
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VDD	VDD&VDDRF	Power Supply (3V)
VSS	VSS&VSSRF	Ground

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

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

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

In Application Programming – IAP

The device offers IAP function to update data or application program to Flash ROM. Users can define any ROM location for IAP, but there are some features which user must notice in using IAP function.

BC68F2130 Configurations		BC68F2140 Configurations	
Erase Block	256 words / block	Erase Block	256 words / block
Writing Word	4 words / time	Writing Word	4 words / time
Reading Word	1 word / time	Reading Word	1 word / time

In Application Programming Control Registers

The Address registers, FARL and FARH, and the Data registers, FD0L/FD0H, FD1L/FD1H, FD2L/FD2H, FD3L/FD3H, together with the Control registers, FC0 and FC1, located in Data Memory Sector 0, are the corresponding Flash access registers for IAP. All read and write operations to the registers can be performed using the direct addressing access.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FARL	A7	A6	A5	A4	A3	A2	A1	A0
FARH (BC68F2130)	—	—	—	—	—	A10	A9	A8
FARH (BC68F2140)	—	—	—	—	A11	A10	A9	A8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Registers List

- **FC0 Register**

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

Bit 7 **CFWEN**: Flash Memory Write enable control
0: Flash Memory write function is disabled
1: Flash Memory write function has been successfully enabled
When this bit is cleared to 0 by application program, the Flash Memory write function is disabled. Note that writing a "1" into this bit results in no action. This bit is used to indicate that the Flash Memory write function status. When this bit is set to 1 by hardware, it means that the Flash Memory write function is enabled successfully. Otherwise, the Flash Memory write function is disabled as the bit content is zero.

Bit 6~4 **FMOD2~FMOD0**: Mode selection
000: Write Program Memory
001: Block erase Program Memory
010: Reserved
011: Read Program Memory
100: Reserved
101: Reserved
110: FWEN mode – Flash Memory write function enable mode
111: Reserved

Bit 3 **FWPEN**: Flash Memory Write procedure enable control
0: Disable
1: Enable
When this bit is set to 1 and the FMOD field is set to "110", the IAP controller will execute the "Flash memory write function enable" procedure. Once the Flash memory write function is successfully enabled, it is not necessary to set the FWPEN bit any more.

Bit 2 **FWT**: Flash ROM write control bit
0: Do not initiate Flash Memory write or Flash Memory write process is completed
1: Initiate Flash Memory write process
This bit can be set by software only, when the write process is completed, hardware will clear the FWT bit.

Bit 1 **FRDEN**: Flash Memory read enabled bit
0: Flash Memory read disable
1: Flash Memory read enable

Bit 0 **FRD**: Flash Memory read control bit
0: Do not initiate Flash Memory read or Flash Memory read process is completed
1: Initiate Flash Memory read process
This bit can be set by software only, when the read process is completed, hardware will clear the FRD bit.

- **FC1 Register**

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

Bit 7~0 **D7~D0**: whole chip reset
When user writes a specific value of "55H" to this register, it will generate a reset signal to reset whole chip.

- **FARL Register**

Bit	7	6	5	4	3	2	1	0
Name	A7	A6	A5	A4	A3	A2	A1	A0
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 **A7~A0**: Flash Memory Address [7:0]

- **FARH Register – BC68F2130**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	A10	A9	A8
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"
Bit 2~0 **A10~A8**: Flash Memory Address [10:8]

- **FARH Register – BC68F2140**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	A11	A10	A9	A8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"
Bit 3~0 **A11~A8**: Flash Memory Address [11:8]

- **FD0L Register**

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

Bit 7~0 **D7~D0**: The first Flash Memory data [7:0]

- **FD0H Register**

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

Bit 7~0 **D15~D8**: The first Flash Memory data [15:8]

- **FD1L Register**

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

Bit 7~0 **D7~D0**: The second Flash Memory data [7:0]

- **FD1H Register**

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

Bit 7~0 **D15~D8**: The second Flash Memory data [15:8]

- **FD2L Register**

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

Bit 7~0 **D7~D0**: The third Flash Memory data [7:0]

- **FD2H Register**

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

Bit 7~0 **D15~D8**: The third Flash Memory data [15:8]

- **FD3L Register**

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

Bit 7~0 **D7~D0**: The fourth Flash Memory data [7:0]

- **FD3H Register**

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

Bit 7~0 **D15~D8**: The fourth Flash Memory data [15:8]

Flash Memory Write Function Enable Procedure

In order to allow users to change the Flash Memory data through the IAP control registers, users must first enable the Flash Memory write operation by the following procedure:

Step 1. Write "110" into the FMOD2~FMOD0 bits to select the FWEN mode.

Step 2. Set the FWPEN bit to "1". The step 1 and step 2 can be executed simultaneously.

Step 3. The pattern data with a sequence of 00H, 04H, 0DH, 09H, C3H and 40H must be written into the FD1L, FD1H, FD2L, FD2H, FD3L and FD3H registers respectively.

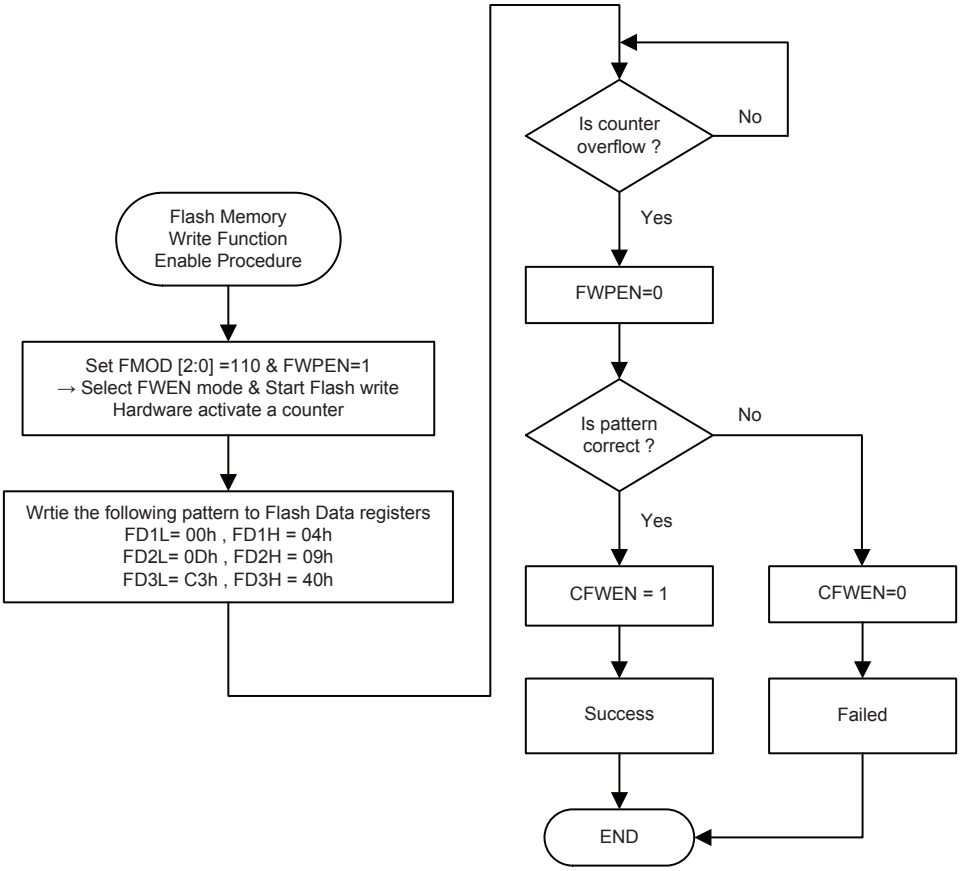
Step 4. A counter with a time-out period of 300μs will be activated to allow users writing the correct pattern data into the FD1L/FD1H~FD3L/FD3H register pairs. The counter clock is derived from the f_{SUB} clock.

Step 5. If the counter overflows or the pattern data is incorrect, the Flash Memory write operation will not be enabled and users must again repeat the above procedure. Then the FWPEN bit will be automatically cleared to 0 by hardware.

Step 6. If the pattern data is correct before the counter overflows, the Flash Memory write operation will be enabled and the FWPEN bit will automatically be cleared to 0 by hardware. The CFWEN bit will also be set to 1 by hardware to indicate that the Flash Memory write operation is successfully enabled.

Step 7. Once the Flash Memory write operation is enabled, the user can change the Flash Memory data through the Flash control register.

Step 8. To disable the Flash Memory write operation, the user can clear the CFWEN bit to 0.



Flash Memory Write Function Enable Procedure

Flash Memory Read/Write Procedure

After the Flash Memory write function is successfully enabled through the preceding IAP procedure, users must first erase the corresponding Flash Memory block and then initiate the Flash Memory write operation. For the devices the number of the block erase operation is 256 words per block, the available block erase address is only specified by FARH register and the content in the FARL register is not used to specify the block address.

Erase Block	FARH [2:0]	FARL [7:0]
0	0000 0000	xxxx xxxx
1	0000 0001	xxxx xxxx
2	0000 0010	xxxx xxxx
3	0000 0011	xxxx xxxx
4	0000 0100	xxxx xxxx
5	0000 0101	xxxx xxxx
6	0000 0110	xxxx xxxx
7	0000 0111	xxxx xxxx

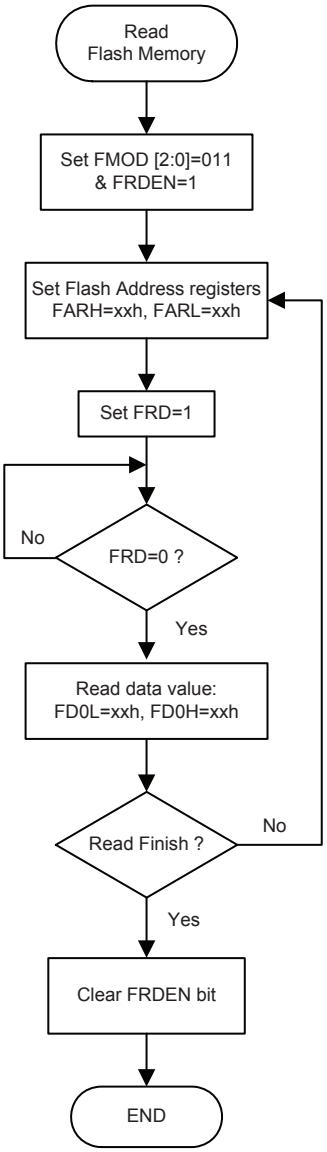
"x": don't care

BC68F2130 Erase Block Number and Selection

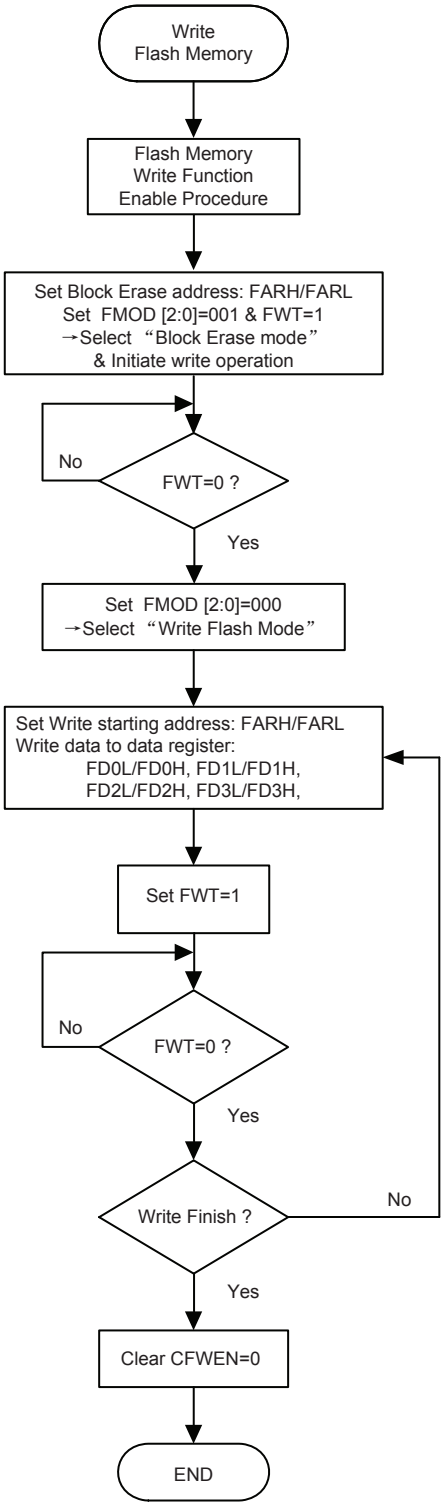
Erase Block	FARH [3:0]	FARL [7:0]
0	0000 0000	xxxx xxxx
1	0000 0001	xxxx xxxx
2	0000 0010	xxxx xxxx
3	0000 0011	xxxx xxxx
4	0000 0100	xxxx xxxx
5	0000 0101	xxxx xxxx
6	0000 0110	xxxx xxxx
7	0000 0111	xxxx xxxx
8	0000 1000	xxxx xxxx
9	0000 1001	xxxx xxxx
10	0000 1010	xxxx xxxx
11	0000 1011	xxxx xxxx
12	0000 1100	xxxx xxxx
13	0000 1101	xxxx xxxx
14	0000 1110	xxxx xxxx
15	0000 1111	xxxx xxxx

"x": don't care

BC68F2140 Erase Block Number and Selection



Read Flash Memory Procedure



Write Flash Memory Procedure

Note: When the FWT or FRD bit is set to 1, the MCU is stopped.

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 devices. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

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

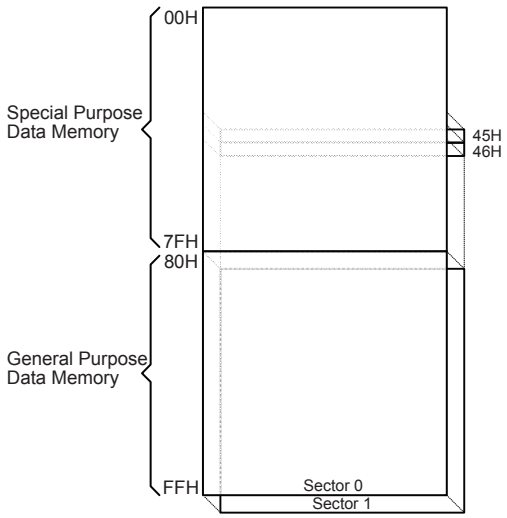
Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide Memory. Each of the Data Memory Sectors 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.

Device	Special Purpose Data Memory	General Purpose Data Memory	
	Located Sectors	Capacity	Sector: Address
BC68F2130 BC68F2140	0, 1	256×8	0: 80H~FFH 1: 80H~FFH

Data Memory Summary



Data Memory Structure

Data Memory Addressing

For the devices that support 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.

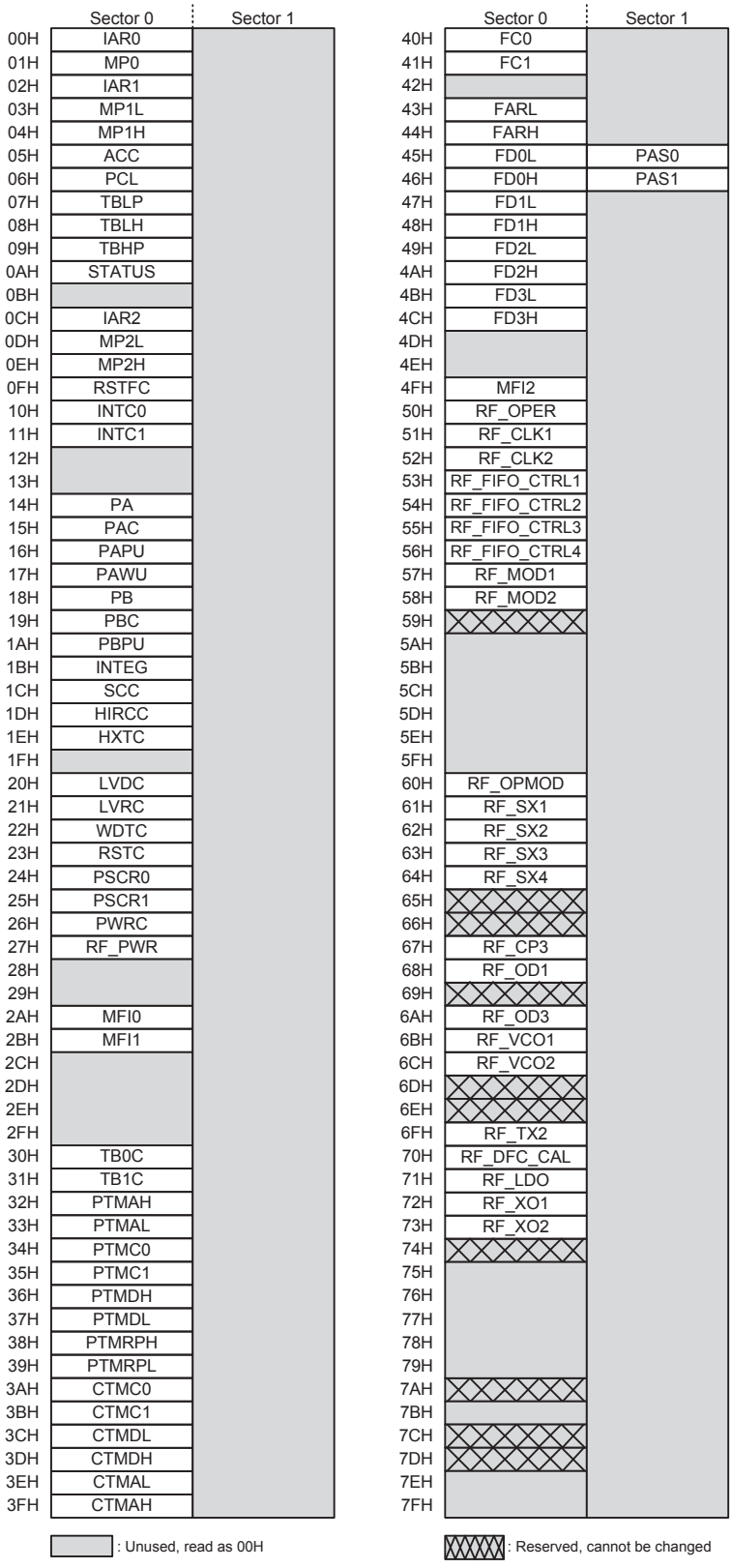
Direct Addressing can be used in all sectors using the extended instructions which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address "m" in the extended instructions has 9 valid bits for these devices, 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

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.

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 1

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

Indirect Addressing Program 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 registers are important and if the subroutine can corrupt the status register, precautions must be taken to correctly save it.

STATUS Register

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

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 selection and operation are selected through the relevant control registers.

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for other functions such as the RF circuits, the Watchdog Timer and Time Base Interrupts. An external oscillator requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options are selected through the relevant control registers. 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 devices have the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

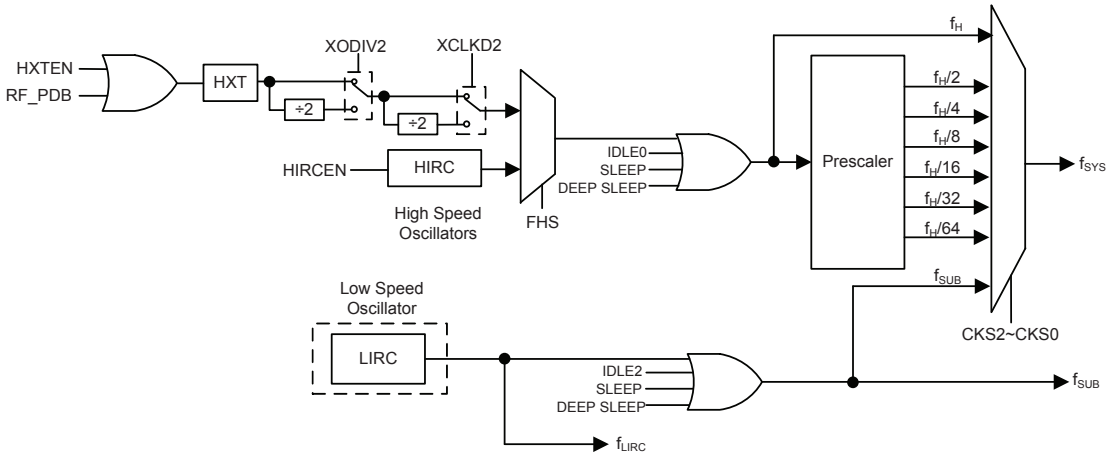
Type	Name	Frequency	Pins
External High Speed Crystal	HXT	12/12.8/16/19.2MHz	OSC1/OSC2
Internal High Speed RC	HIRC	16MHz	—
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

System Clock Configurations

There are several oscillator sources, two high speed oscillators and one low speed oscillator. The high speed system clocks are sourced from an external crystal/ceramic oscillator, HXT, and an 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.

The actual source clock used for the high speed oscillator source clock is selected by the FHS bit in the SCC register. The frequency of the slow speed or high speed system clock is also determined using the CKS2~CKS0 bits in the SCC register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.

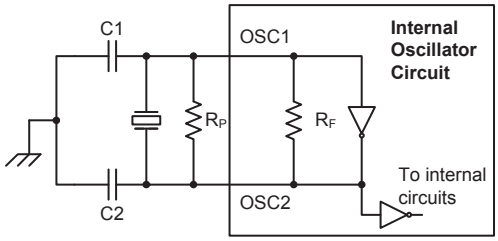


System Clock Configurations

External Crystal/Ceramic Oscillator – HXT

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

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



Note: 1. R_P is normally not required. C1 and C2 are required.
2. The capacitor load is internally integrated and determined by the RF_XO1 register.

Crystal/Resonator Oscillator

Crystal Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
16MHz	0pF	0pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has 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. Note that if this internal system clock option is selected, as it requires no external pins for its operation.

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 at 5V, 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

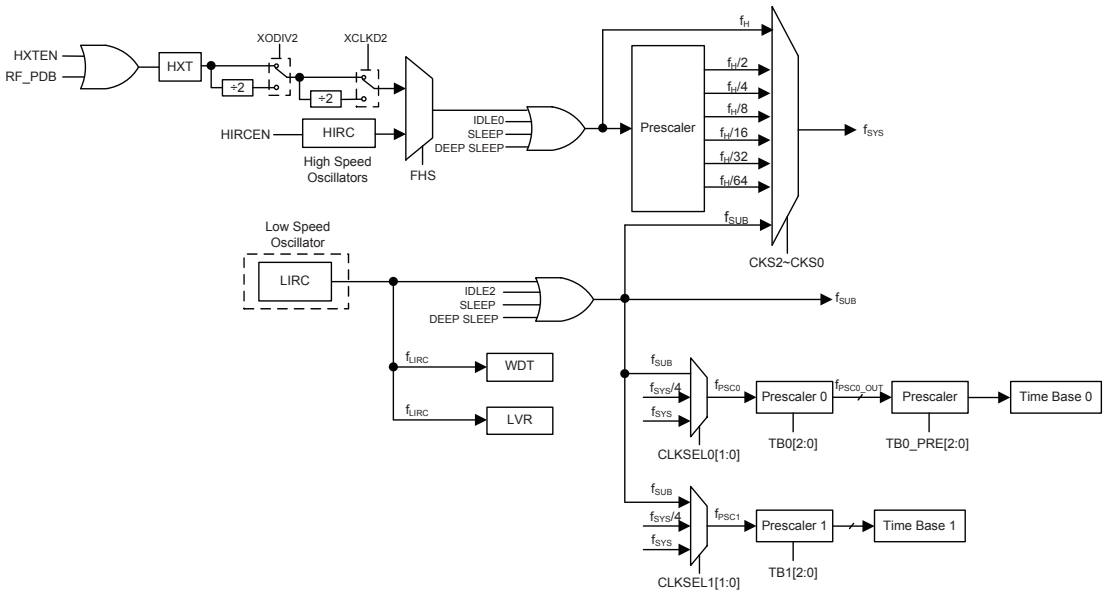
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their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

System Clocks

The devices have 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 speed system clock can be sourced from an HXT or HIRC oscillator, selected via configuring the FHS bit in the SCC register. The HXT oscillator is enabled using the HXTEN and RF_PDB bits. Its frequency can be divided by two or four using the XODIV2 and XCLKD2 bits. These latter three bits are described in the RF register section.

The low speed system clock source can be sourced from the internal clock f_{SUB}. If f_{SUB} is selected then it is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.



Device Clock Configurations

Note: 1. 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, $f_H \sim f_H/64$, for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

2. When the system enters the DEEP SLEEP Mode, the Time Base 0 clock source is f_{LIRC}, which is determined by the WDT on/off status.

System Operation Modes

There are seven different modes of operation for the microcontrollers, 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 NORMAL Mode and SLOW Mode. The remaining five modes, the DEEP SLEEP, SLEEP, IDLE0, IDLE1 and IDLE2 Mode are used when the microcontroller CPU is switched off to conserve power.

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

"x": Don't care

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

2. The f_{LIRC} clock can be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP or DEEP SLEEP mode.

NORMAL Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational, where the 1.5V LDO is turned on with the PWDN bit in the PWRC register being low and the system clock is provided by one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source sourced from one of the high speed oscillators, either the HXT or HIRC oscillators. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The 1.5V LDO is turned on with the PWDN bit in the PWRC register being low. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from the LIRC oscillator.

SLEEP Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the FHIDEN and FSIDEN bit are low, the PWDN bit in the PWRC register is low. In the SLEEP mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral function will also be stopped, too. However the f_{LIRC} clock can continues to operate if the WDT function is enabled.

IDLE0 Mode

The IDLE0 Mode is entered when an HALT instruction is executed and when the FHIDEN bit in the SCC register is low, the FSIDEN bit in the SCC register is high and the the PWDN bit in the PWRC register is low. 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 an HALT instruction is executed and when the FHIDEN bit in the SCC register is high, the FSIDEN bit in the SCC register is high and the the PWDN bit in the PWRC register is low. 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 an HALT instruction is executed and when the FHIDEN bit in the SCC register is high, the FSIDEN bit in the SCC register is low and the the PWDN bit in the PWRC 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.

DEEP SLEEP Mode

The DEEP SLEEP Mode is entered when an HALT instruction is executed and when the PWDN bit in the PWRC register is high. In the DEEP 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 can continue to operate if the WDT function is enabled.

Control Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	FHS	—	FHIDEN	FSIDEN
HIRCC	—	—	—	—	—	—	HIRCF	HIRCEN
HXTC	—	—	—	—	—	—	HXTF	HXTEN
PWRC	PA_WAKE	—	—	TB0_WAKE	POF33V	LCMD	IO_ISO_EN	PWDN

System Operating Mode Control Registers List

SCC Register

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

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

Bit 4 Unimplemented, read as "0"

Bit 3 **FHS**: High Frequency clock selection
0: HIRC – internal high frequence oscillator
1: HXT – external crystal
When this bit is set high to select the external HXT oscillator as the system clock source, the actual clock input to the MCU is determined by the crystal oscillator connected to the OSC1 and OSC2 pins together with the XODIV2 bit in the RF_XO2 register and the XCLKD2 bit in the RF_CLK1 register. For example, if f_{HXT}=16MHz, XODIV2=0 and XCLKD2=0, then f_H=16MHz.

Bit 2 Unimplemented, read as "0"

Bit 1 **FHIDEN**: High Frequency oscillator control when the 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 an "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 an "HALT" instruction. The LIRC oscillator is controlled by this bit together with the WDT function enable control when the LIRC is selected to be the low speed oscillator clock source or the WDT function is enabled respectively. If this bit is cleared to 0 but the WDT function is enabled, the f_{LIRC} clock will also be enabled.

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 to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

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

HXTC Register

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

Bit 7~2 Unimplemented, read as "0"

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

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

PWRC Register

Bit	7	6	5	4	3	2	1	0
Name	PA_WAKE	—	—	TB0_WAKE	POF33V	LCMD	IO_ISO_EN	PWDN
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	1	0	0	0

Bit 7 **PA_WAKE**: Port A wake-up MCU from DEEP SLEEP Mode flag
0: No Port A wake-up MCU from DEEP SLEEP Mode occured
1: Port A wake-up MCU from DEEP SLEEP Mode occured
This bit indicates whether the MCU has been woken up from the DEEP SLEEP Mode by the port A wake-up function. The PAWU register should be properly configured before the MCU enters the DEEP SLEEP Mode.

Bit 6~5 Unimplemented, read as "0"

Bit 4 **TB0_WAKE**: Time Base 0 wake-up MCU from DEEP SLEEP Mode flag
0: No Time Base 0 wake-up MCU from DEEP SLEEP Mode occured
1: Time Base 0 wake-up MCU from DEEP SLEEP Mode occured
This bit indicates whether the MCU has been woken up from the DEEP SLEEP Mode by the Time Base 0 interrupt. The TB0C register and the TB0F interrupt flag should be properly configured before the MCU enters the DEEP SLEEP Mode.

Bit 3 **POF33V**: 3V power domain power on reset flag
0: No 3V power domain power on reset occured
1: 3V power domain power on reset occured
This bit is set high by hardware when a 3V power domain power on reset occurs. It is cleared by application program or when the "CLR WDT" instruction is executed.

Bit 2 **LCMD**: 1.5V LDO low current mode selection
0: 1.5V LDO normal mode
1: 1.5V LDO low current mode

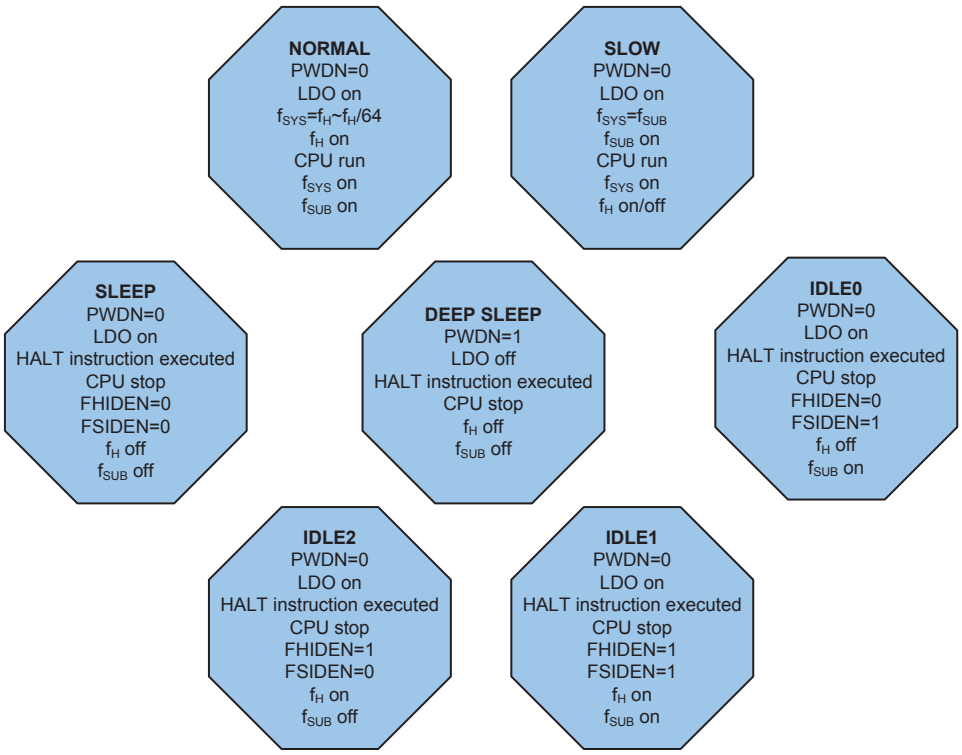
Bit 1 **IO_ISO_EN**: I/O isolation mode selection
0: I/O in normal operation mode
1: I/O in isolation mode (I/O status remains unchanged)
Before entering the DEEP SLEEP Mode this bit must be set high to latch I/O ports, so that I/O ports status will remain unchanged when in the DEEP SLEEP Mode. After the MCU is woken up from the DEEP SLEEP Mode, this bit must be cleared using application program to de-latch I/O ports.

Bit 0 **PWDN**: DEEP SLEEP Mode control
0: MCU under normal operation
1: MCU enters the DEEP SLEEP Mode after HALT (1.5V LDO off by hardware)
If this bit is set high, after executing the HALT instruction, the MCU will enter the DEEP SLEEP Mode, in which case the 1.5V LDO will be turned off by hardware. Before entering the DEEP SLEEP Mode, users should previously store the 1.5V domain system settings to Data Memory and latch the I/O ports using the application program.

Operating Mode Switching

The devices 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.

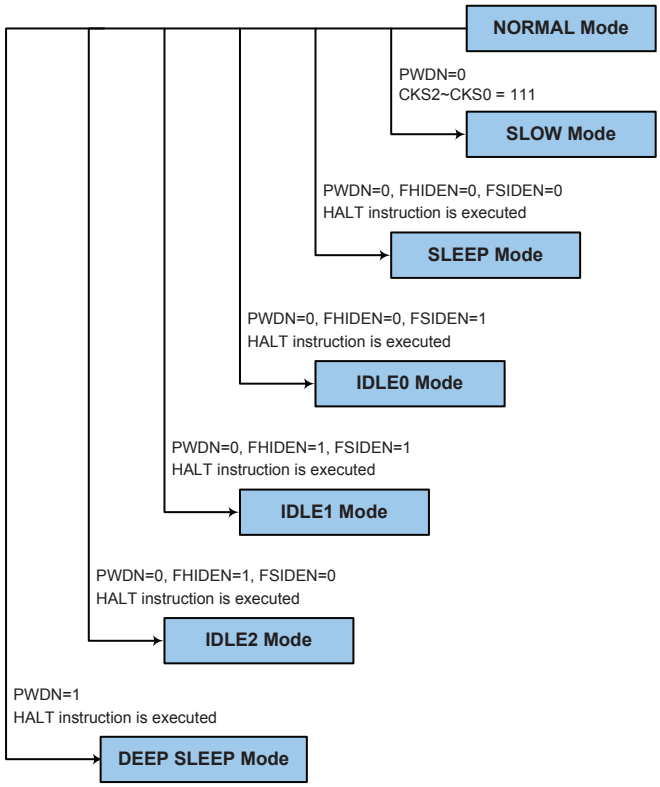
In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE or DEEP SLEEP Modes is executed via the HALT instruction. When an HALT instruction is executed, whether the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



NORMAL Mode to SLOW Mode Switching

When running in the NORMAL 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 set 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.

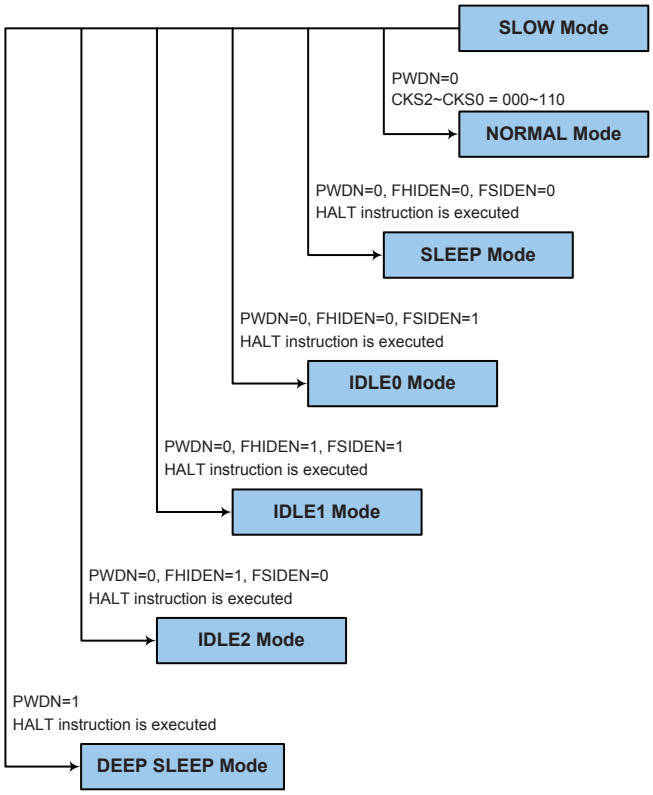
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to NORMAL Mode Switching

In SLOW mode the system clock is derived from f_{SUB}. When system clock is switched back to the NORMAL 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 NORMAL mode from the SLOW mode. This is monitored using the HXTF bit in the HXTC register or the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the A.C. Characteristics.



Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

There is only one way for the devices to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with the PWDN bit in the PWRC register equal to "0" and both the FHIDEN and FSIDEN bits in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

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

Entering the IDLE2 Mode

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

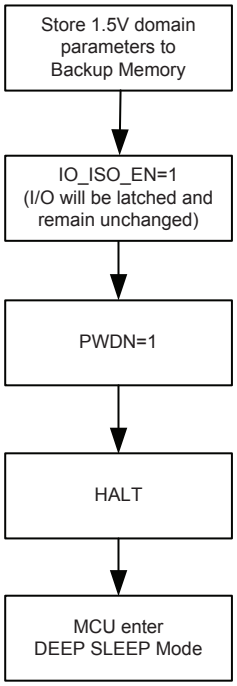
- The f_H clock will be on but the f_{SUB} clock will be off and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down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

Entering the DEEP SLEEP Mode

There is only one way for the devices to enter the DEEP SLEEP Mode and that is to execute the "HALT" instruction in the application program with the PWDN bit in the PWRC register equal to "1". 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. It is recommended to previously store the important 1.5V domain system settings in the Data Memory.
- The I/O ports will maintain their present conditions if the IO_ISO_EN bit in the PWRC register is set high.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

The accompany flowchart shows the program procedure for entering the DEEP SLEEP Mode.



Standby Current Considerations

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

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

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

Wake-up

To minimise power consumption the devices can enter the DEEP SLEEP, SLEEP or any IDLE Modes, 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

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

- An external falling edge on Port A
- A Time Base 0 interrupt

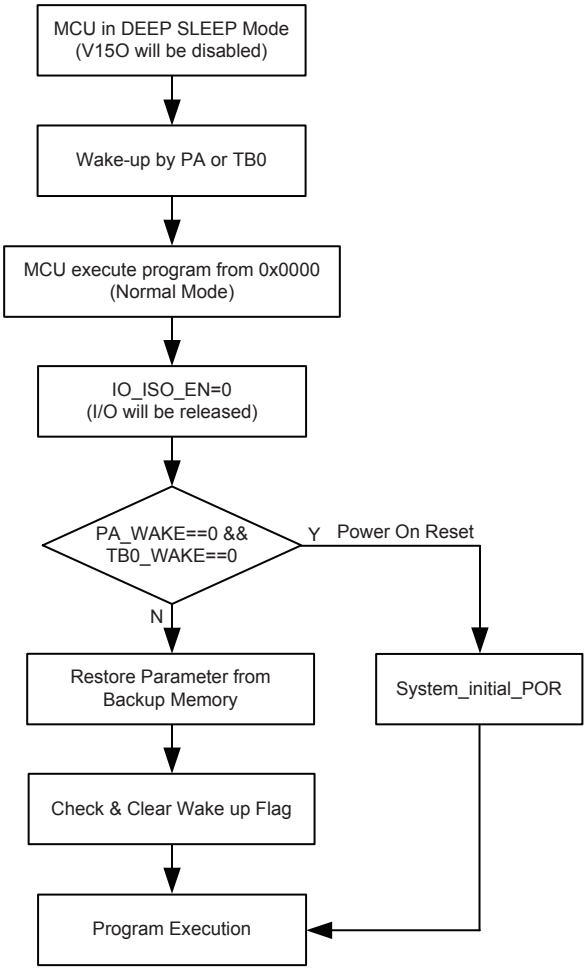
When the device executes the “HALT” instruction, the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction. If the system is woken up by a WDT overflow, a Watchdog Timer reset will be initiated and the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

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a pin wakes-up the system from the SLEEP or IDLE Mode, the program will resume execution at the instruction following the “HALT” instruction. However, when a pin wakes-up the system from the DEEP SLEEP Mode, the program will resume execution from 0000h.

If the system is woken up from the SLEEP or IDLE Mode 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 the system is woken up from the DEEP SLEEP Mode by the Time Base 0 interrupt, the program will resume execution from 0000h.

If an interrupt request flag is set high before entering the DEEP SLEEP, SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

The accompany flowchart shows the program procedure for waking up from the DEEP SLEEP Mode.



Watchdog Timer

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

Watchdog Timer Clock Source

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

Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period as well as the enable/disable operation.

WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT function software control

If the WDT is always enabled determined by the configuration option:
10101: Enable
01010: Enable
Other values: Reset MCU

If the WDT is controlled by the WDTC register determined by the configuration option:
10101: Disable
01010: Enable
Other values: Reset MCU

The WE4~WE0 bits setup is determined by the WDT configuration option selection.

When these bits are changed to any other values except 10101B and 01010B 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 timeout period.

RSTFC Register

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

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

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

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

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

Watchdog Timer Operation

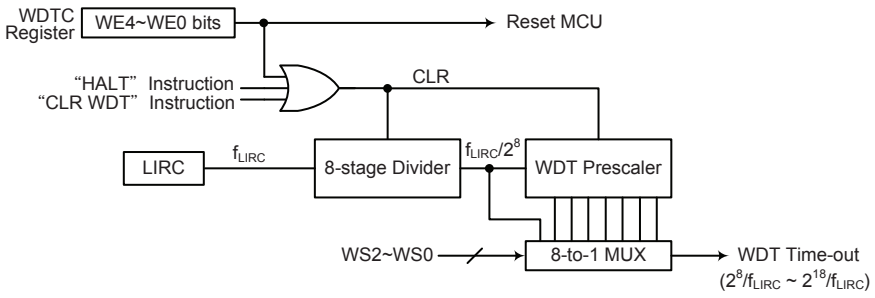
The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control and reset control of the Watchdog Timer. The WDT function is determined by the related configuration option together with the WE4~WE0 bits, as summerised in the following table. 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.

Configuration Option	WE4~WE0 Bits	WDT Function
WDT always enabled	10101B or 01010B	Enable
	Any other value	Reset MCU
WDT controlled by WDTC	10101B	Disable
	01010B	Enable
	Any other value	Reset MCU

Watchdog Timer Enable/Disable 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 DEEP SLEEP, SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bits, the second is using the Watchdog Timer software clear instruction, 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 second for the 2^{18} division ratio, and a minimum timeout of 8ms for the 2^8 division ration.



Watchdog Timer

Reset and Initialisation

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

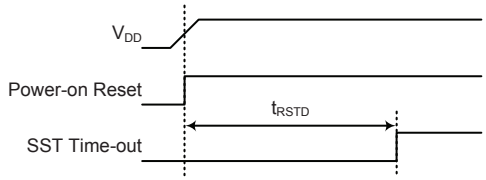
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

Reset Functions

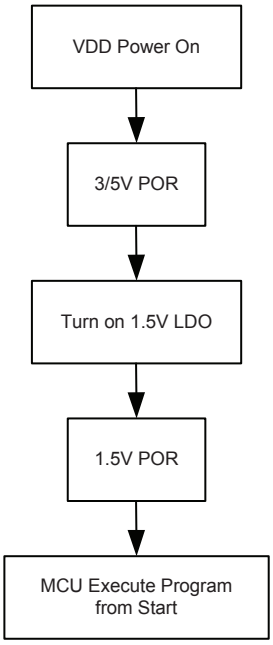
There are several ways in which a microcontroller reset can occur, through events occurring internally.

Power-on Reset

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



Power-on Reset Timing Chart



Internal Reset Control

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

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

Internal Reset Function Control

- **RSTC Register**

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

Bit 7~0 **RSTC7~RSTC0**: Reset function control
01010101: On operation
10101010: On operation
Other values: Reset MCU

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

- **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

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

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

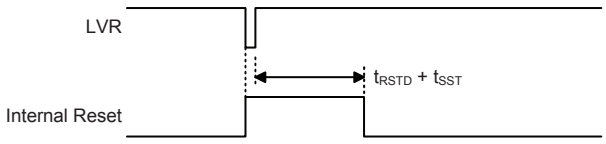
Bit 1 **LRF**: LVR control register software reset flag
Refer to the Low Voltage Reset section.

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

Low Voltage Reset – LVR

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

The LVR function is enabled/disabled using the related configuration option with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVD/LVR characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value is fixed at 1.9V. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the DEEP SLEEP, SLEEP or IDLE mode.



Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR Voltage Select control
01010101: 1.9V
00110011: 1.9V
10011001: 1.9V
10101010: 1.9V
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. In this situation the register contents will remain the same after such a reset occurs.

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

- **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

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

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

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

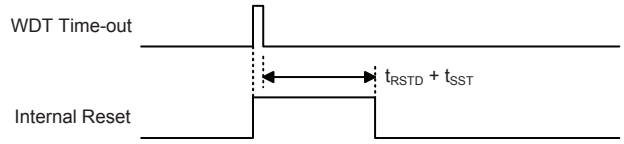
Bit 0 **WRF**: WDT control register software reset flag
Refer to the Watchdog Timer Control Register section.

IAP Reset

The IAP reset is caused by writing data 55H to the FC1 register.

Watchdog Time-out Reset during Normal Operation

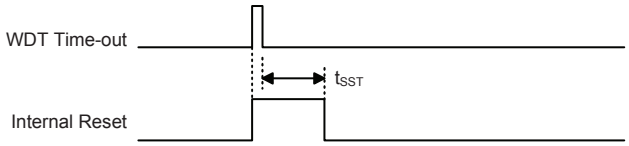
The Watchdog time-out Reset during normal operations is the same as a LVR reset except that the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during DEEP SLEEP/SLEEP/IDLE Mode

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



WDT Time-out Reset during DEEP SLEEP/SLEEP/IDLE 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 NORMAL or SLOW Mode operation
1	u	WDT time-out reset during NORMAL or SLOW Mode operation
1	1	WDT time-out reset during DEEP SLEEP, IDLE or SLEEP Mode operation

"u" stands for unchanged

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

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

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

Register	BC68F2130	BC68F2140	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
IAR0	●	●	0000 0000	0000 0000	uuuu uuuu
MP0	●	●	0000 0000	0000 0000	uuuu uuuu
IAR1	●	●	0000 0000	0000 0000	uuuu uuuu
MP1L	●	●	0000 0000	0000 0000	uuuu uuuu
MP1H	●	●	0000 0000	0000 0000	uuuu uuuu
ACC	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	●	●	0000 0000	0000 0000	0000 0000
TBLP	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	●		---- -xxx	---- -uuu	---- -uuu
		●	---- xxxx	---- uuuu	---- uuuu
STATUS	●	●	xx00 xxxx	xx1u 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	●	●	---- 0x00	---- uuuu	---- uuuu
INTC0	●	●	-000 0000	-000 0000	-uuu uuuu
INTC1	●	●	0000 0000	0000 0000	uuuu uuuu
PA	●		--11 1111	--11 1111	--uu uuuu
		●	1111 1111	1111 1111	uuuu uuuu
PAC	●		--11 1111	--11 1111	--uu uuuu
		●	1111 1111	1111 1111	uuuu uuuu
PAPU	●		--00 0000	--00 0000	--uu uuuu
		●	0000 0000	0000 0000	uuuu uuuu
PAWU	●		--00 0000	--00 0000	--uu uuuu
		●	0000 0000	0000 0000	uuuu uuuu
PB	●		--11 ----	--11 ----	--uu ----
		●	--11 1111	--11 1111	--uu uuuu
PBC	●		--11 ----	--11 ----	--uu ----
		●	--11 1111	--11 1111	--uu uuuu
PBPU	●		--00 ----	--00 ----	--uu ----
		●	--00 0000	--00 0000	--uu uuuu
INTEG	●	●	---- 0000	---- 0000	---- uuuu
SCC	●	●	001- 0x00	001- 0x00	uuu- uxuu
HIRCC	●	●	---- xx01	---- xx01	---- xxuu
HXTC	●	●	---- --00	---- --00	---- --uu
LVDC	●	●	--00 x000	--00 x000	--uu xuuu
LVRC	●	●	0101 0101	0101 0101	uuuu uuuu
WDTC	●	●	0101 0011	0101 0011	uuuu uuuu
RSTC	●	●	0101 0101	0101 0101	uuuu uuuu
PSCR0	●	●	---- --00	---- --00	---- --uu
PSCR1	●	●	---- --00	---- --00	---- --uu
PWRC	●	●	0--0 1000	u--u uuuu	u--u uuuu
RF_PWR	●	●	---- ---0	---- ---0	---- ---u
MFI0	●	●	--00 --00	--00 --00	--uu --uu

Register	BC68F2130	BC68F2140	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
MFI1	●	●	--00 --00	--00 --00	--uu --uu
TB0C	●	●	0000 -000	0000 -000	uuuu -uuu
TB1C	●	●	0--- -000	0--- -000	u--- -uuu
PTMAH	●	●	---- --00	---- --00	---- --uu
PTMAL	●	●	0000 0000	0000 0000	uuuu uuuu
PTMC0	●	●	0000 0---	0000 0---	uuuu u---
PTMC1	●	●	0000 0000	0000 0000	uuuu uuuu
PTMDH	●	●	---- --00	---- --00	---- --uu
PTMDL	●	●	0000 0000	0000 0000	uuuu uuuu
PTMRPH	●	●	---- --00	---- --00	---- --uu
PTMRPL	●	●	0000 0000	0000 0000	uuuu uuuu
CTMC0	●	●	0000 0000	0000 0000	uuuu uuuu
CTMC1	●	●	0000 0000	0000 0000	uuuu uuuu
CTMDL	●	●	0000 0000	0000 0000	uuuu uuuu
CTMDH	●	●	---- --00	---- --00	---- --uu
CTMAL	●	●	0000 0000	0000 0000	uuuu uuuu
CTMAH	●	●	---- --00	---- --00	---- --uu
FC0	●	●	0000 0000	0000 0000	uuuu uuuu
FC1	●	●	0000 0000	0000 0000	uuuu uuuu
FARL	●	●	0000 0000	0000 0000	uuuu uuuu
FARH	●		---- -000	---- -000	---- -uuu
		●	---- 0000	---- 0000	---- uuuu
FD0L	●	●	--00 0000	--00 0000	--uu uuuu
FD0H	●	●	0000 0000	0000 0000	uuuu uuuu
FD1L	●	●	0000 0000	0000 0000	uuuu uuuu
FD1H	●	●	0000 0000	0000 0000	uuuu uuuu
FD2L	●	●	0000 0000	0000 0000	uuuu uuuu
FD2H	●	●	0000 0000	0000 0000	uuuu uuuu
FD3L	●	●	0000 0000	0000 0000	uuuu uuuu
FD3H	●	●	0000 0000	0000 0000	uuuu uuuu
MFI2	●	●	-000 -000	-000 -000	-uuu -uuu
RF_OPER	●	●	--00 ---0	--00 ---0	--uu ---u
RF_CLK1	●	●	10-0 ---0	10-0 ---0	uu-u ---u
RF_CLK2	●	●	0000 0000	0000 0000	uuuu uuuu
RF_FIFO_CTRL1	●	●	0000 0000	0000 0000	uuuu uuuu
RF_FIFO_CTRL2	●	●	0011 1111	0011 1111	uuuu uuuu
RF_FIFO_CTRL3	●	●	0000 0000	0000 0000	uuuu uuuu
RF_FIFO_CTRL4	●	●	0--- 0001	0--- 0001	u--- uuuu
RF_MOD1	●	●	1001 0000	1001 0000	uuuu uuuu
RF_MOD2	●	●	---- -001	---- -001	---- -uuu
RF_OPMOD	●	●	---- -000	---- -000	---- -uuu
RF_SX1	●	●	-011 0110	-011 0110	-uuu uuuu
RF_SX2	●	●	0000 1010	0000 1010	uuuu uuuu
RF_SX3	●	●	1101 0111	1101 0111	uuuu uuuu
RF_SX4	●	●	---- 0011	---- 0011	---- uuuu

Register	BC68F2130	BC68F2140	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
RF_CP3	•	•	1100 1010	1100 1010	uuuu uuuu
RF_OD1	•	•	0000 0100	0000 0100	uuuu uuuu
RF_OD3	•	•	--01 -100	--01 -100	--uu -uuu
RF_VCO1	•	•	0001 0000	0001 0000	uuuu uuuu
RF_VCO2	•	•	0-10 0000	0-10 0000	u-uu uuuu
RF_TX2	•	•	1101 -000	1101 -000	uuuu -uuu
RF_DFC_CAL	•	•	00-0 0000	00-0 0000	uu-u uuuu
RF_LDO	•	•	00011000	00011000	uuuu uuuu
RF_XO1	•	•	10-1 0101	10-1 0101	uu-u uuuu
RF_XO2	•	•	-01- 0011	-01- 0011	-uu- uuuu
PAS0	•	•	0000 0000	0000 0000	uuuu uuuu
PAS1	•		---- 0000	---- 0000	---- uuuu
		•	--00 0000	--00 0000	--uu uuuu

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

Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA~PB. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. 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	—	—	PA5	PA4	PA3	PA2	PA1	PA0
PAC	—	—	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	—	—	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	—	—	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	PB5	PB4	—	—	—	—
PBC	—	—	PBC5	PBC4	—	—	—	—
PBPU	—	—	PBPU5	PBPU4	—	—	—	—

“—”: Unimplemented, read as “0”

I/O Logic Function Registers List – BC68F2130

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	—	—	PB5	PB4	PB3	PB2	PB1	PB0
PBC	—	—	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0

"—": Unimplemented, read as "0"

I/O Logic Function Registers List – BC68F2140

Pull-high Resistors

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

PxPU Register

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

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

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

Port A Wake-up

The HALT instruction forces the microcontroller into the DEEP SLEEP, 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-shared functional pin is selected as general purpose input/output and the MCU enters the DEEP SLEEP, SLEEP or IDLE 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

PAWUn: Port A Pin wake-up function control
0: Disable
1: Enable
The PAWUn bit is used to control the Port A pin wake-up function. However, the actual available bits for each device may be different.

I/O Port Control Registers

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

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 and B. However, the actual available bits for each I/O Port of each device 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 devices include Port "x" Output Function Selection register "n", labeled as PxSn, 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, special point must be noted for some digital input pins, such as INTn, xTCKn, xTPnI, etc, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the revelant pin-shared control bit fields. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup forementioned, they must also be setup as input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1 (BC68F2130)	—	—	—	—	PAS13	PAS12	PAS11	PAS10
PAS1 (BC68F2140)	—	—	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10

Pin-shared Function Selection Registers List

- **PAS0 Register – BC68F2130/BC68F2140**

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

Bit 7~6 **PAS07~PAS06**: PA3 Pin-Shared function selection
00/01/10/11: PA3/INT0

Bit 5~4 **PAS05~PAS04**: PA2 Pin-Shared function selection
00/10/11: PA2
01: PTP

Bit 3~2 **PAS03~PAS02**: PA1 Pin-Shared function selection
00/01/10/11: PA1/PTCK

Bit 1~0 **PAS01~PAS00**: PA0 Pin-Shared function selection
00/10/11: PA0/PTPI
01: PTPB

- **PAS1 Register – BC68F2130**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **PAS13~PAS12**: PA5 Pin-Shared function selection
00/10/11: PA5
01: CTP

Bit 1~0 **PAS11~PAS10**: PA4 Pin-Shared function selection
00/01/10/11: PA4/INT1

- **PAS1 Register – BC68F2140**

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

Bit 7~6 Unimplemented, read as "0"

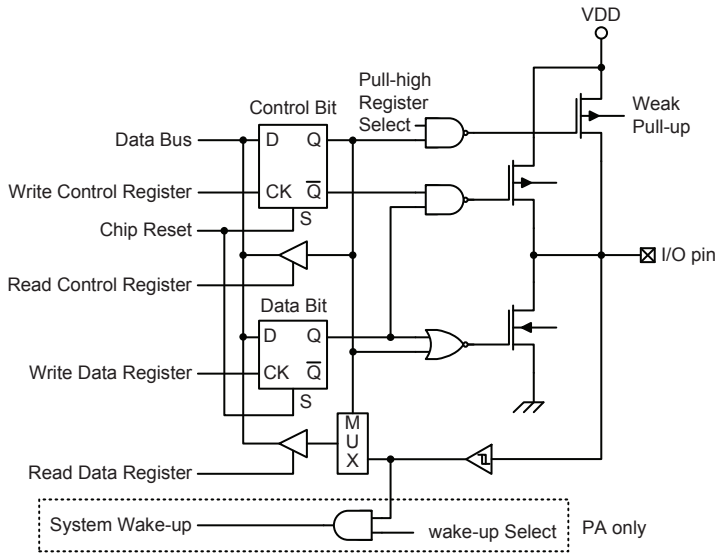
Bit 5~4 **PAS15~PAS14**: PA6 Pin-Shared function selection
00/10/11: PA6
01: CTPB

Bit 3~2 **PAS13~PAS12**: PA5 Pin-Shared function selection
00/10/11: PA5
01: CTP

Bit 1~0 **PAS11~PAS10**: PA4 Pin-Shared function selection
00/01/10/11: PA4/INT1

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

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

Timer Modules – TM

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

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

Introduction

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

Function	CTM	PTM
Timer/Counter	√	√
Input Capture	—	√
Compare Match Output	√	√
PWM Channels	1	1
Single Pulse Output	—	1
PWM Alignment	Edge	Edge
PWM Adjustment Period & Duty	Duty or Period	Duty or Period

TM Function Summary

CTM	PTM
10-bit CTM	10-bit PTM

TM Name/Type Reference

TM Operation

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

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTCK2~xTCK0 bits in the xTM control registers, where "x" stands for C or P type TM. The clock source can be a ratio of the system clock f_{SYS} or the internal high clock f_H, the f_{SUB} clock source or the external xTCK pin. The xTCK pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

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

TM External Pins

Each of the TMs, irrespective of what type, has one TM input pin, with the label xTCK. The xTM input pin, xTCK, is essentially a clock source for the xTM and is selected using the xTCK2~xTCK0 bits in the xTMC0 register. This external TM input pin allows an external clock source to drive the internal TM. The xTCK input pin can be chosen to have either a rising or falling active edge. The PTCK pin is also used as the external trigger input pin in single pulse output mode.

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

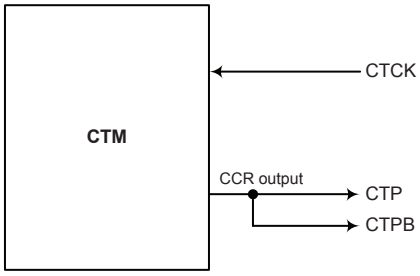
The TMs each have two output pins with the label xTP and xTPB. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTP and xTPB output pins are also the pins where the TM generates the PWM output waveform. As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using the relevant pin-shared function selection bits described in the Pin-shared Function section.

CTM		PTM	
Input	Output	Input	Output
CTCK	CTP, CTPB	PTCK, PTPI	PTP, PTPB

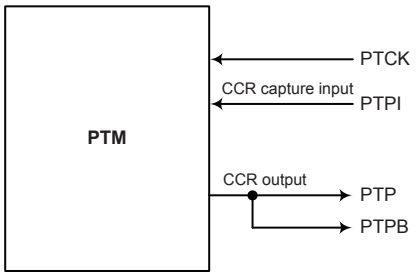
TM External Pins

TM Input/Output Pin Selection

Selecting to have a TM input/output or whether to retain its other shared function is implemented using the relevant pin-shared function selection registers, with the corresponding selection bits in each pin-shared function register corresponding to a TM input/output pin. Configuring the selection bits correctly will setup the corresponding pin as a TM input/output. The details of the pin-shared function selection are described in the pin-shared function section.



CTM Function Pin Control Block Diagram

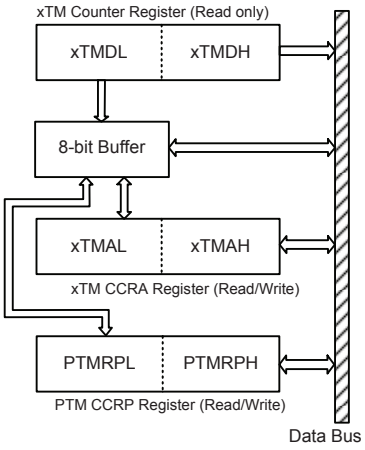


PTM Function Pin Control Block Diagram

Programming Considerations

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

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

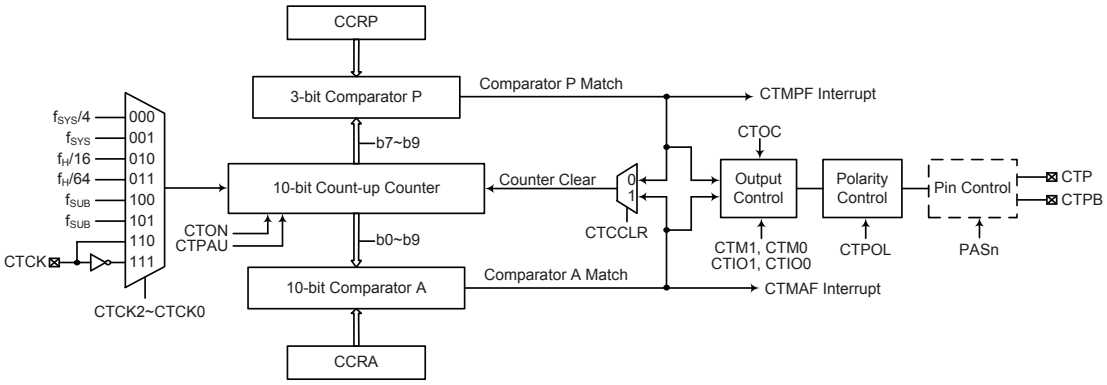


The following steps show the read and write procedures:

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

Compact Type TM – CTM

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



Compact Type TM Block Diagram

Compact TM Operation

At its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is three bits wide whose value is compared with the highest three bits in the counter while the CCRA is the ten bits and therefore compares with all counter bits.

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

Compact Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
CTMC0	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
CTMC1	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
CTMDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMDH	—	—	—	—	—	—	D9	D8
CTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMAH	—	—	—	—	—	—	D9	D8

10-bit Compact TM Register List

CTMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
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 **CTPAU**: CTM Counter Pause Control
0: Run
1: Pause

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

Bit 6~4 **CTCK2~CTCK0**: Select CTM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: CTCK rising edge clock
111: CTCK falling edge clock

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

Bit 3 **CTON**: CTM Counter On/Off Control
0: Off
1: On

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

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

Bit 2~0 **CTRP2~CTRP0**: CTM CCRP 3-bit register, compared with the CTM Counter bit 9~bit 7 Comparator P Match Period
000: 1024 CTM clocks
001: 128 CTM clocks
010: 256 CTM clocks
011: 384 CTM clocks
100: 512 CTM clocks
101: 640 CTM clocks
110: 768 CTM clocks
111: 896 CTM clocks

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

CTMC1 Register

Bit	7	6	5	4	3	2	1	0
Name	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **CTM1~CTM0**: Select CTM Operating Mode
00: Compare Match Output Mode
01: Undefined
10: PWM Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the CTM. To ensure reliable operation the CTM should be switched off before any changes are made to the CTM1 and CTM0 bits. In the Timer/Counter Mode, the CTM output pin state is undefined.

Bit 5~4 **CTIO1~CTIO0**: Select CTP output function

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

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

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the CTIO1 and CTIO0 bits determine how the CTM output pin changes state when a compare match occurs from the Comparator A. The CTM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the CTM output pin should be setup using the CTOC bit in the CTMC1 register. Note that the output level requested by the CTIO1 and CTIO0 bits must be different from the initial value setup using the CTOC bit otherwise no change will occur on the CTM output pin when a compare match occurs. After the CTM output pin changes state it can be reset to its initial level by changing the level of the CTON bit from low to high.

In the PWM Output Mode, the CTIO1 and CTIO0 bits determine how the CTM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the CTIO1 and CTIO0 bits only after the CTM has been switched off. Unpredictable PWM outputs will occur if the CTIO1 and CTIO0 bits are changed when The CTM is running.

Bit 3 **CTOC**: CTP Output control bit

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

PWM Output Mode
0: Active low
1: Active high

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

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

Bit 1 **CTDPX**: CTM PWM period/duty Control
0: CCRP - period; CCRA - duty
1: CCRP - duty; CCRA - period
This bit, determines which of the CCRA and CCRP registers are used for period and duty control of the PWM waveform.

Bit 0 **CTCCLR**: Select CTM Counter clear condition
0: CTM Comparatror P match
1: CTM Comparatror A match
This bit is used to select the method which clears the counter. Remember that the Compact TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the CTCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The CTCCLR bit is not used in the PWM Output Mode.

CTMDL 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**: CTM Counter Low Byte Register bit 7 ~ bit 0
CTM 10-bit Counter bit 7 ~ bit 0

CTMDH Register

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

Bit 7~2 Unimplemented, read as “0”

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

CTMAL 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**: CTM CCRA Low Byte Register bit 7 ~ bit 0
CTM 10-bit CCRA bit 7 ~ bit 0

CTMAH Register

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

Bit 7~2 Unimplemented, read as "0"

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

Compact Type TM Operating Modes

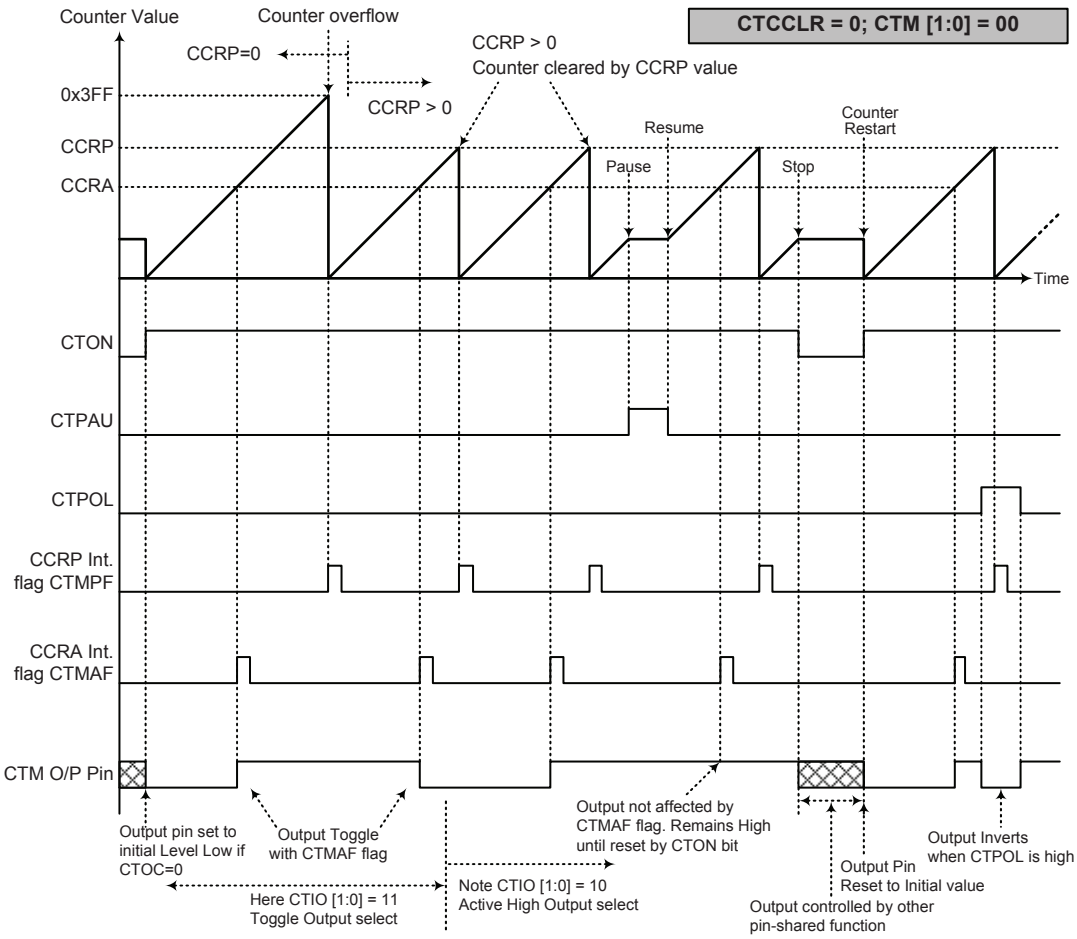
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTM1 and CTM0 bits in the CTMC1 register.

Compare Match Output Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMAF and CTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

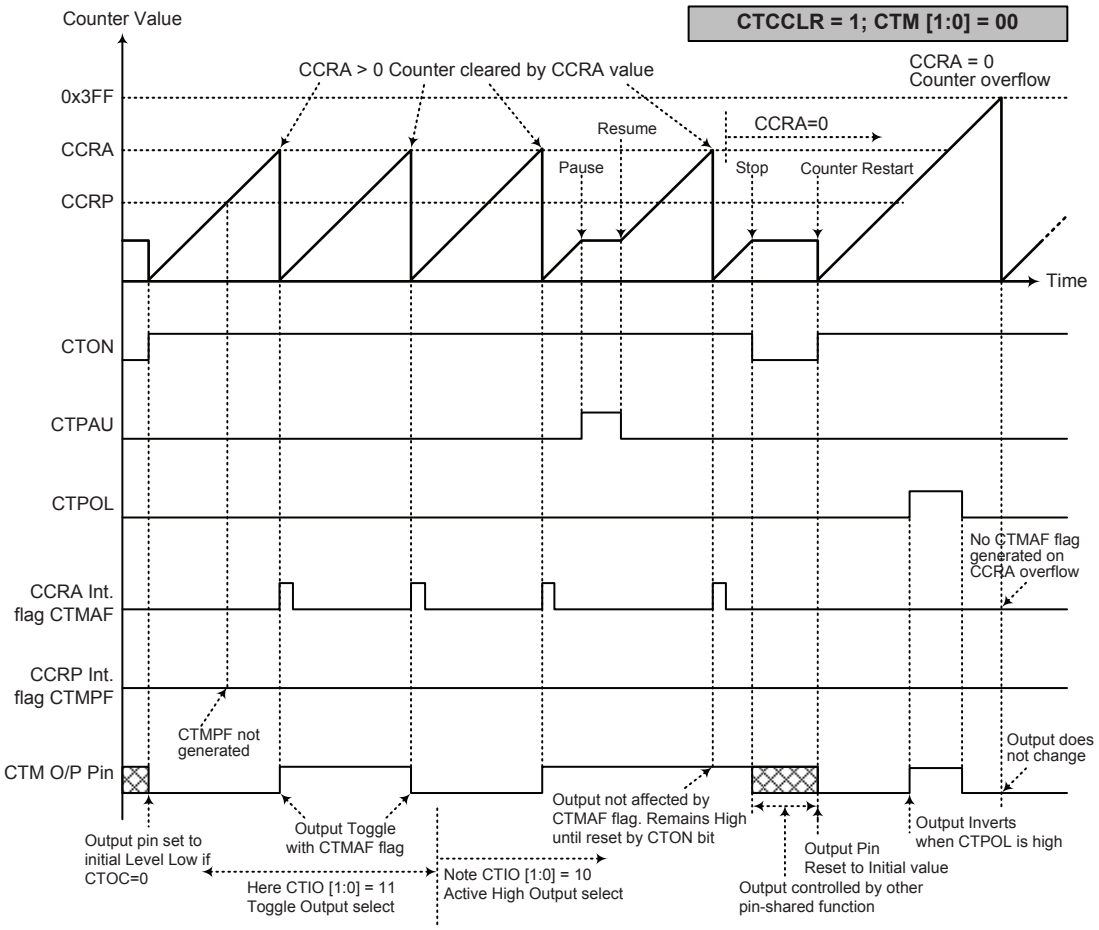
If the CTCCLR bit in the CTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTCCLR is high no CTMPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the CTMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the CTM output pin will change state. The CTM output pin condition however only changes state when a CTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTM output pin. The way in which the CTM output pin changes state are determined by the condition of the CTIO1 and CTIO0 bits in the CTMC1 register. The CTM output pin can be selected using the CTIO1 and CTIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the CTM output pin, which is setup after the CTON bit changes from low to high, is setup using the CTOC bit. Note that if the CTIO1 and CTIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – CTCCLR=0

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



Compare Match Output Mode – CTCCLR=1

Note: 1. With CTCCLR = 1, a Comparator A match will clear the counter
2. The CTM output pin controlled only by the CTMAF flag
3. The output pin reset to initial state by a CTON rising edge
4. The CTMPF flags is not generated when CTCCLR = 1

Timer/Counter Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the CTM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the CTM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 register should be set to 10 respectively. The PWM function within the CTM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the CTM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the CTCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTDPX bit in the CTMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The CTOC bit In the CTMC1 register is used to select the required polarity of the PWM waveform while the two CTIO1 and CTIO0 bits are used to enable the PWM output or to force the CTM output pin to a fixed high or low level. The CTPOL bit is used to reverse the polarity of the PWM output waveform.

- **CTM, PWM Output Mode, Edge-aligned Mode, CTDPX=0**

CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	128	256	384	512	640	768	896	1024
Duty	CCRA							

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

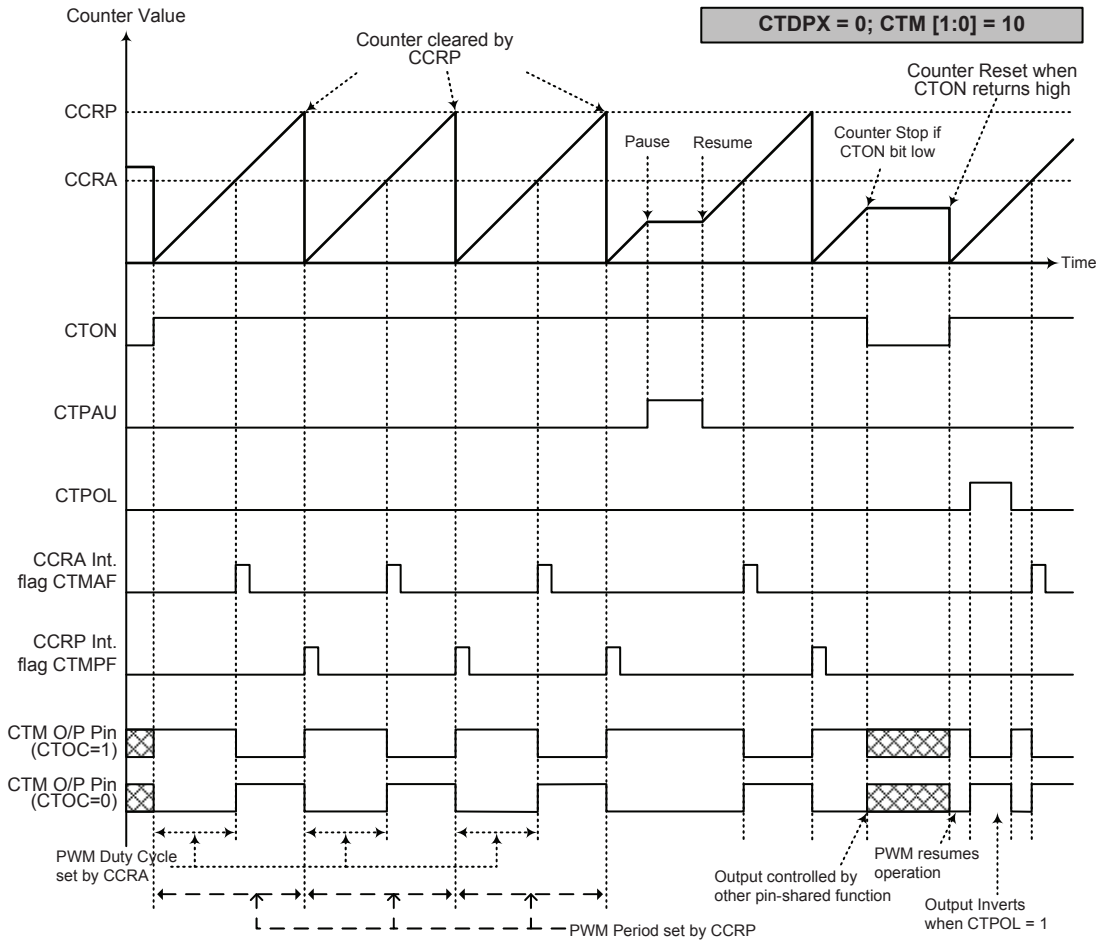
The CTM PWM output frequency=(f_{SYS}/4)/512=f_{SYS}/2048=3.9063kHz, duty=128/512=25%.

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

- **CTM, PWM Output Mode, Edge-aligned Mode, CTDPX=1**

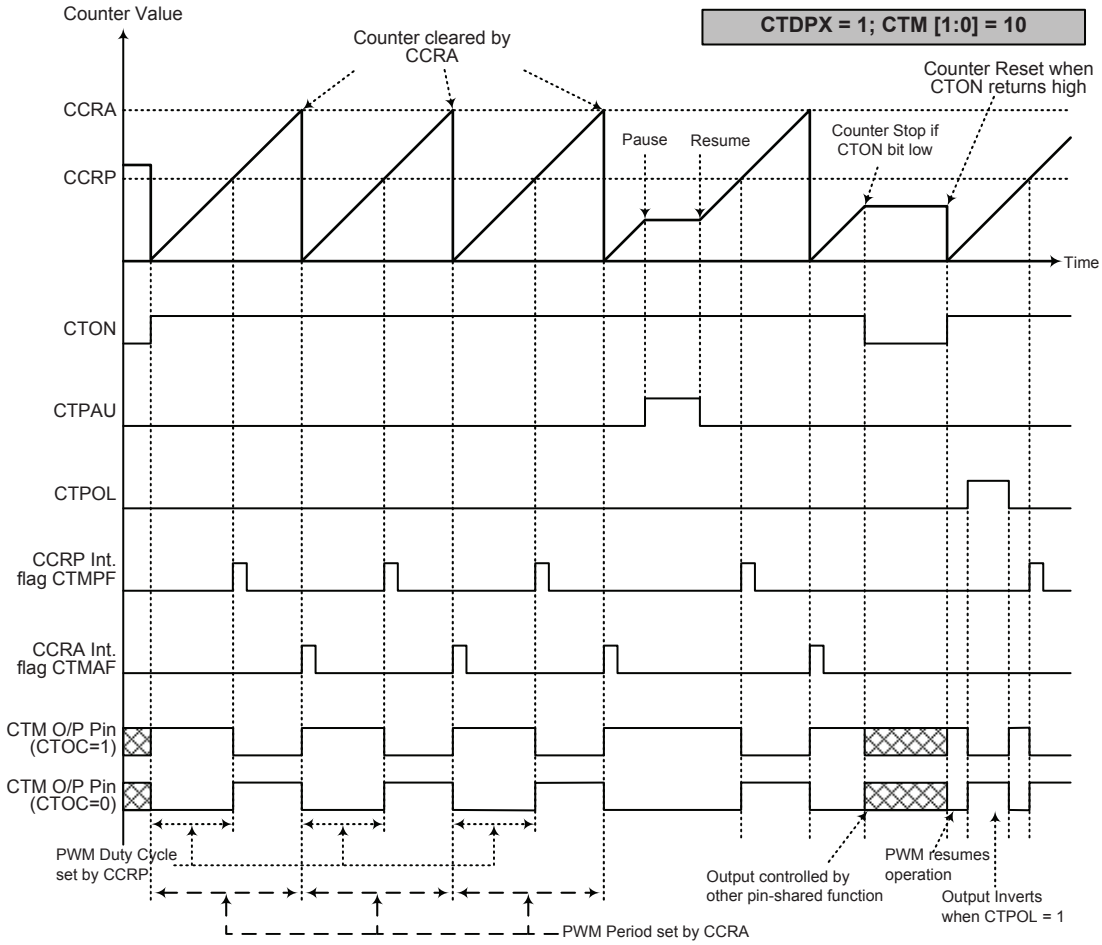
CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

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



PWM Output Mode – CTDPX=0

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

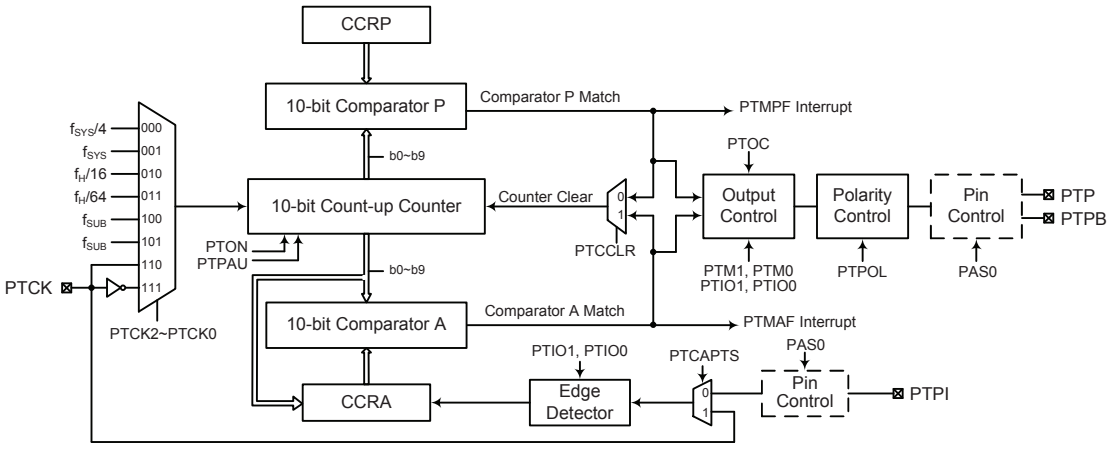


PWM Output Mode – CTDPX=1

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

Periodic Type TM – PTM

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



Periodic Type TM Block Diagram

Periodic TM Operation

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

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

Periodic Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	—	—	—	—	—	—	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	—	—	—	—	—	—	D9	D8
PTMRPL	PTRP7	PTRP6	PTRP5	PTRP4	PTRP3	PTRP2	PTRP1	PTRP0
PTMRPH	—	—	—	—	—	—	PTRP9	PTRP8

10-bit Periodic TM Registers List

PTMC0 Register

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

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

Bit 6~4 **PTCK2~PTCK0**: Select PTM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: PTCK rising edge clock
111: PTCK falling edge clock
These three bits are used to select the clock source for the PTM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the oscillator section.

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

Bit 2~0 Unimplemented, read as “0”

PTMC1 Register

Bit	7	6	5	4	3	2	1	0
Name	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PTM1~PTM0**: Select PTM Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

Bit 5~4 **PTIO1~PTIO0**: Select PTM external pin PTP, PTPI or PTCK function

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

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

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

Timer/Counter Mode
Unused

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

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

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

Bit 3 **PTOC**: PTM PTP Output control bit

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

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

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

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

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

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

Bit 0 **PTCCLR**: Select PTM Counter clear condition
0: PTM Comparator P match
1: PTM Comparator A match

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

PTMDL Register

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

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

PTMDH Register

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

Bit 7~2 Unimplemented, read as “0”

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

PTMAL Register

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

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

PTMAH Register

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

Bit 7~2 Unimplemented, read as "0"

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

PTMRPL Register

Bit	7	6	5	4	3	2	1	0
Name	PTRP7	PTRP6	PTRP5	PTRP4	PTRP3	PTRP2	PTRP1	PTRP0
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 **PTRP7~PTRP0**: PTM CCRP Low Byte Register bit 7 ~ bit 0
PTM 10-bit CCRP bit 7 ~ bit 0

PTMRPH Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PTRP9	PTRP8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PTRP9~PTRP8**: PTM CCRP High Byte Register bit 1 ~ bit 0
PTM 10-bit CCRP bit 9 ~ bit 8

Periodic Type TM Operating Modes

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

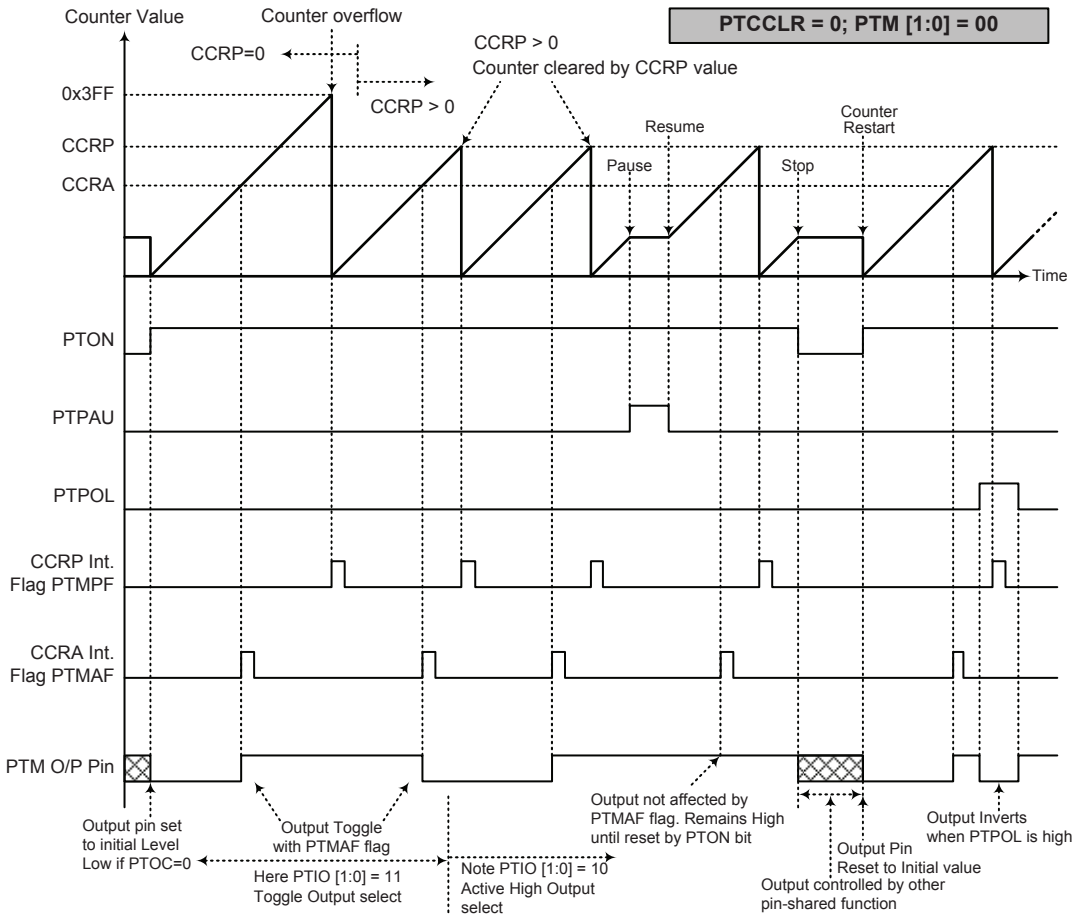
Compare Match Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the PTCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both PTMAF and PTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

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

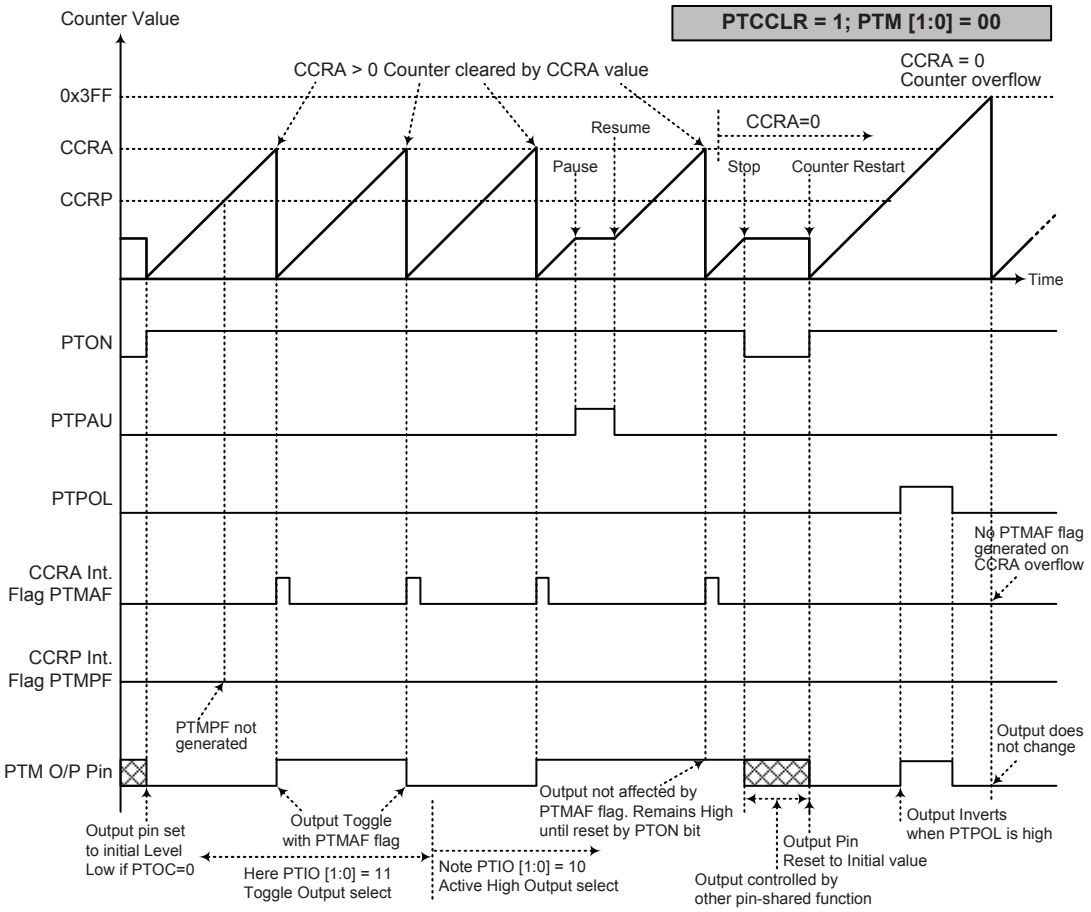
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the PTMAF interrupt request flag will not be generated.

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



Compare Match Output Mode – PTCCLR = 0

Note: 1. With PTCCLR=0 a Comparator P match will clear the counter
2. The PTM output pin is controlled only by the PTMAF flag
3. The output pin is reset to its initial state by a PTON bit rising edge



Compare Match Output Mode – PTCCLR = 1

Note: 1. With PTCCLR=1 a Comparator A match will clear the counter
2. The PTM output pin is controlled only by the PTMAF flag
3. The output pin is reset to its initial state by a PTON bit rising edge
4. A PTMPF flag is not generated when PTCCLR=1

Timer/Counter Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the TM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the PTM output pins are not used in this mode, the pins can be used as normal I/O pins or other pin-shared functions.

PWM Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively. The PWM function within the PTM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the PTM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

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

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

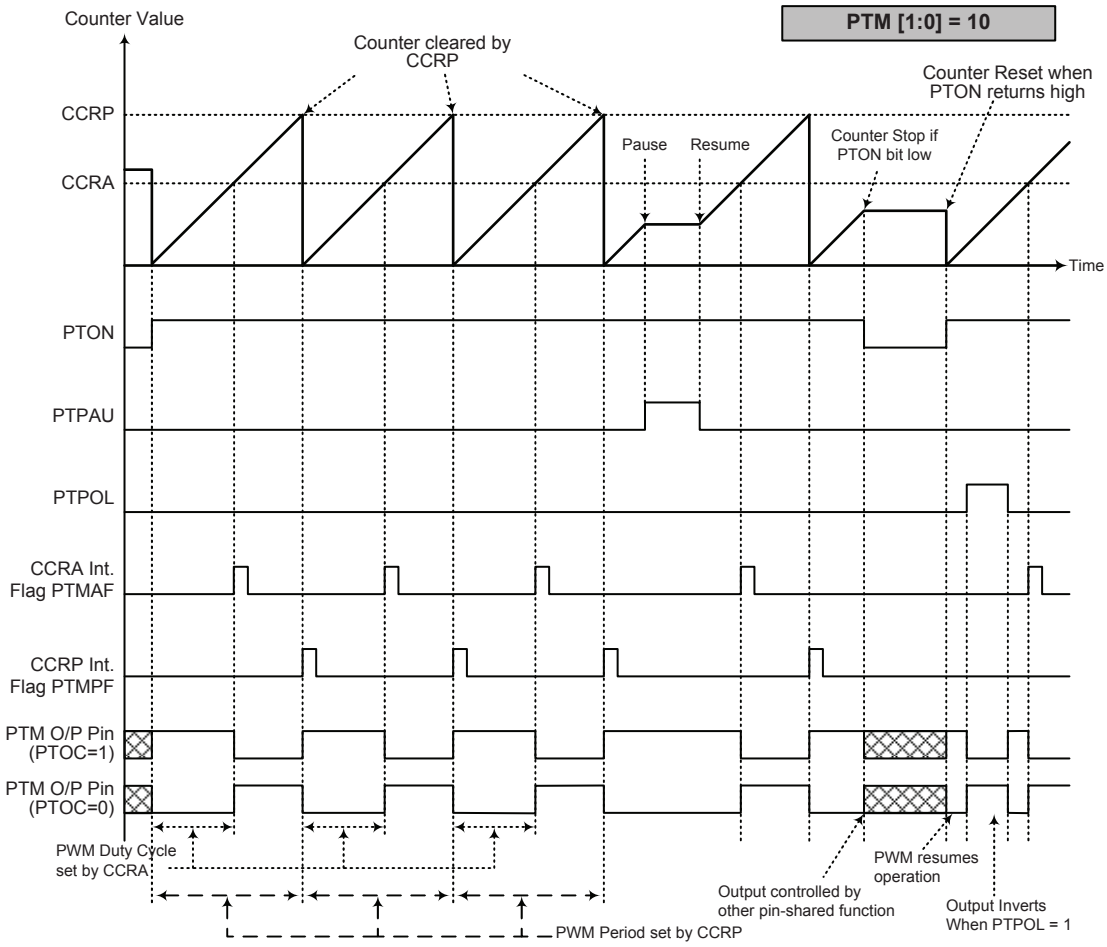
- **10-bit PTM, PWM Output Mode, Edge-aligned Mode**

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

If f_{SYS}=16MHz, PTM clock source select $f_{SYS}/4$, CCRP=512 and CCRA=128,

The PTM PWM output frequency=($f_{SYS}/4$)/512=$f_{SYS}/2048$=7.8125kHz, duty=128/512=25%.

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



PWM Output Mode

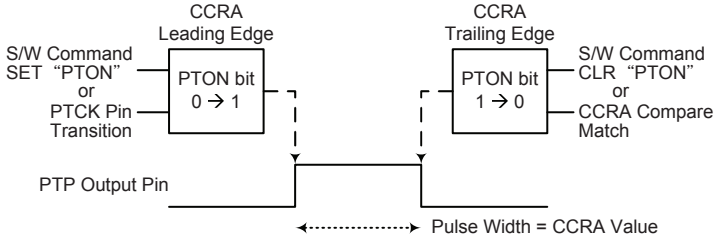
Note: 1. Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTIO[1:0] = 00 or 01
4. The PTCCLR bit has no influence on PWM operation

Single Pulse Output Mode

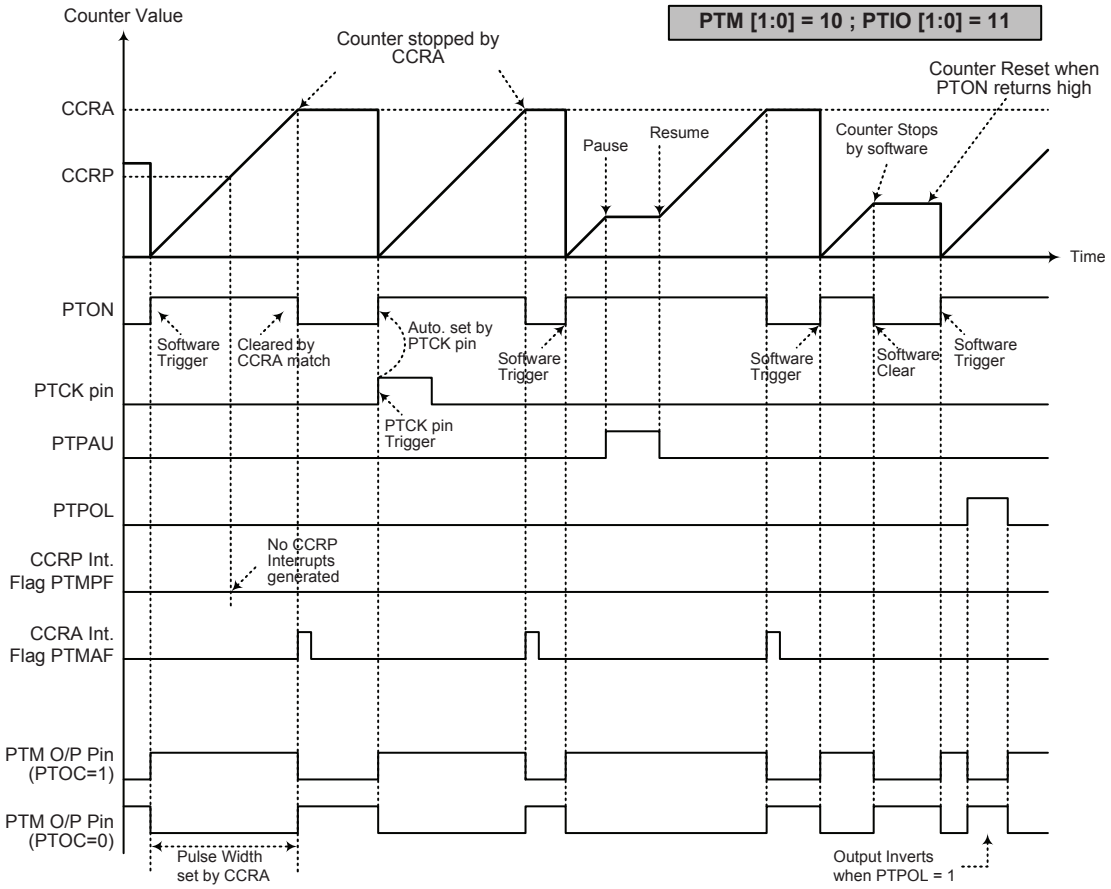
To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

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

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



Single Pulse Generation



Single Pulse Output Mode

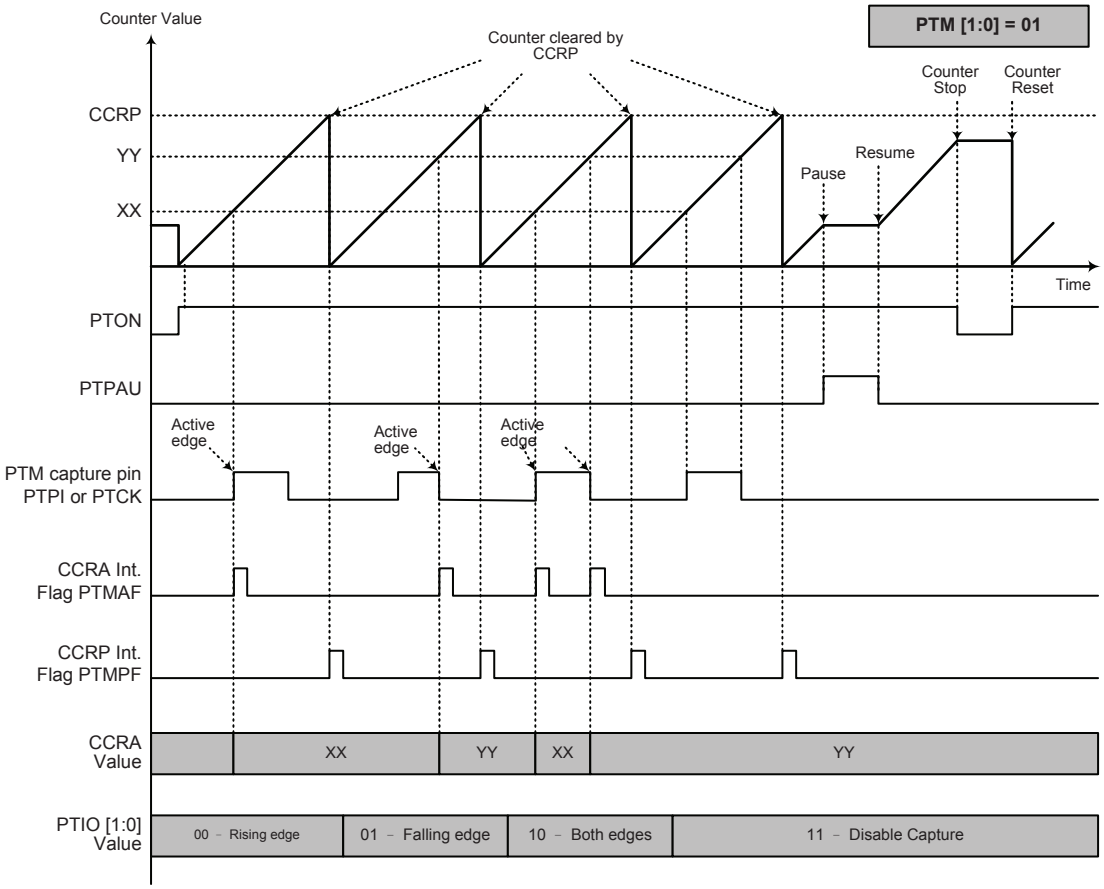
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse is triggered by the PTCK pin or by setting the PTON bit high
4. A PTCK pin active edge will automatically set the PTON bit high
5. In the Single Pulse Output Mode, PTIO[1:0] must be set to “11” and cannot be changed.

Capture Input Mode

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

When the required edge transition appears on the PTPI or PTCK pin the present value in the counter will be latched into the CCRA registers and a PTM interrupt generated. Irrespective of what events occur on the PTPI or PTCK pin, the counter will continue to free run until the PTON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTIO1 and PTIO0 bits can select the active trigger edge on the PTPI or PTCK pin to be a rising edge, falling edge or both edge types. If the PTIO1 and PTIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPI or PTCK pin, however it must be noted that the counter will continue to run.

As the PTPI or PTCK pin is pin shared with other functions, care must be taken if the PTM is in the Capture Input Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The PTCCLR, PTOC and PTPOL bits are not used in this Mode.



Capture Input Mode

Note: 1. PTM[1:0] = 01 and active edge set by the PTIO[1:0] bits
2. A PTM Capture input pin active edge transfers the counter value to CCRA
3. PTCCLR bit not used
4. No output function – PTOC and PTPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero.

RF Transmitter

The RF transmitter is a fully integrated transmitter, which is capable of using both Frequency-Shift Keying (FSK) and On-Off Keying (OOK) modulation modes for data streaming. It has two main operating modes, Burst Mode and Direct Mode. The RF transmitter operates in the 315/433/868/915MHz frequency bands.

RF Transmitter Abbreviation Notes

CP: Charge Pump

DFC: Digital Frequency Centering

FIFO: TX FIFO

MMD: Multi-Mode Divider

PAD: Power Amplifier Driver

SX: Synthesizer

TXD: Data from FIFO or DTXD bit, determined by the DIR_EN bit

VCO: Voltage Control Oscillator

XO: External Crystal Output

XCLK: RF circuit main clock. Controlled by configuring the XO_SEL[2:0], XODIV2 and XCLKD2 bits and enabled by setting the XCLK_EN bit high.

XCLK_MCU: MCU system clock. Controlled by configuring the XO_SEL[2:0], XODIV2 and XCLKD2 bits and enabled by setting the HXTEN bit in the HXTC register high.

RF Transmitter Control Registers

The RF transmitter is controlled by a series of registers. These registers control the overall RF function including power down control, operation mode selection, clock division selecton, FIFO data configuration, modulator control, fractional-N synthesizer control, charge pump (CP) control, multi-mode divider (MMD) control, voltage control oscillator (VCO) control, TX power fine tune, VCO DFC calibration, RF LDO control and RF external crystal output, ect.

Register Name	Bit							
	7	6	5	4	3	2	1	0
RF_PWR	—	—	—	—	—	—	—	RF_PDB
RF_OPER	—	—	FSK_EN	DIR_EN	—	—	—	TX_STROBE
RF_CLK1	XCLK_EN	XCLKINV	—	XCLKD2	—	—	—	RST_RF
RF_CLK2	DTR7	DTR6	DTR5	DTR4	DTR3	DTR2	DTR1	DTR0
RF_FIFO_CTRL1	FFDATA7	FFDATA6	FFDATA5	FFDATA4	FFDATA3	FFDATA2	FFDATA1	FFDATA0
RF_FIFO_CTRL2	FFLEN7	FFLEN6	FFLEN5	FFLEN4	FFLEN3	FFLEN2	FFLEN1	FFLEN0
RF_FIFO_CTRL3	RST_TX_FF	FFSA6	FFSA5	FFSA4	FFSA3	FFSA2	FFSA1	FFSA0
RF_FIFO_CTRL4	DTXD	—	—	—	TXFFLT	FFMG_EN	FFMG1	FFMG0
RF_MOD1	FDEV7	FDEV6	FDEV5	FDEV4	FDEV3	FDEV2	FDEV1	FDEV0
RF_MOD2	—	—	—	—	—	FTEV10	FTEV9	FTEV8
RF_OPMOD	—	—	—	—	—	ACAL_EN	TX_EN	SX_EN
RF_SX1	—	DN6	DN5	DN4	DN3	DN2	DN1	DN0
RF_SX2	DK7	DK6	DK5	DK4	DK3	DK2	DK1	DK0
RF_SX3	DK15	DK14	DK13	DK12	DK11	DK10	DK9	DK8
RF_SX4	—	—	—	—	DK19	DK18	DK17	DK16
RF_CP3	DLY_SYN2	DLY_SYN1	DLY_SYN0	D4	D3	D2	D1	D0
RF_OD1	D7	D6	D5	D4	D3	D2	D1	D0
RF_OD3	—	—	DLY_SXPD1	DLY_SXPD0	—	D2	D1	D0
RF_VCO1	VCO_SWHB	D6	D5	D4	D3	D2	D1	D0

Register Name	Bit							
	7	6	5	4	3	2	1	0
RF_VCO2	DFCSF	—	VCO_DFC4	VCO_DFC3	VCO_DFC2	VCO_DFC1	VCO_DFC0	DFC_OW
RF_TX2	CT_PAD3	CT_PAD2	CT_PAD1	CT_PAD0	—	CT_TXLDO1	CT_TXLDO0	D0
RF_DFC_CAL	CT_MMDLDO1	CT_MMDLDO0	—	D4	D3	D2	D1	D0
RF_LDO	D7	D6	D5	D4	D3	D2	D1	D0
RF_XO1	XSHIFT1	XSHIFT0	—	XO_TRIM4	XO_TRIM3	XO_TRIM2	XO_TRIM1	XO_TRIM0
RF_XO2	—	D6	D5	—	XODIV2	XO_SEL2	XO_SEL1	XO_SEL0

RF Transmitter Control Registers List

Most of the RF transmitter control register details will be described in this section, however several registers will be described in their respective other sections.

RF_PWR Register – RF Power Down Control Register

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **RF_PDB**: RF power down control
0: RF Power down mode
1: RF Active mode
This bit determines whether the RF circuit is in the active or power down mode. This bit will be cleared to 0 when a power on reset, a LVR reset or a WDT time-out during normal operation occurs or when the MCU enters the DEEP SLEEP Mode by setting the PWDN bit in the PWRC register to 1 and executing a HALT instruction.

RF_CLK1 Register – RF Clock Control Register 1

Bit	7	6	5	4	3	2	1	0
Name	XCLK_EN	XCLKINV	—	XCLKD2	—	—	—	RST_RF
R/W	R/W	R/W	—	R/W	—	—	—	R/W
POR	1	0	—	0	—	—	—	0

Bit 7 **XCLK_EN**: Clock auto gating for the RF internal digital circuit
0: Disable
1: Enable
Writing data to the FIFO requires enabling XCLK. The XCLK_EN bit should be cleared before the RF circuit enters the power down mode with the RF_PDB bit cleared, this will prevent XCLK clock glitch from occurring and guarantee that the XCLK domain register will not affected by the clock lost.

Bit 6 **XCLKINV**: XCLK clock invert control
0: XCLK clock does not invert
1: XCLK clock inverts

Bit 5 Unimplemented, read as "0"

Bit 4 **XCLKD2**: XCLK clock divided by 2 control
0: XCLK clock is not divided by 2
1: XCLK clock is divided by 2
It is recommended to clear this bit to zero.

Bit 3~1 Unimplemented, read as "0"

Bit 0 **RST_RF**: Reset RF control register
0: Self-clear RF control register is complete
1: Reset RF control register
When this bit is set to 1, all RF associated registers except the RF_PWR register will be reset. It needs one instruction cycle to complete the self-clear enable action. Do not set the RST_RF bit high in two continuous instructions.

RF_CLK2 Register – RF Clock Control Register 2

Bit	7	6	5	4	3	2	1	0
Name	DTR7	DTR6	DTR5	DTR4	DTR3	DTR2	DTR1	DTR0
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 **DTR7~DTR0**: RF data rate setting

RF data rate = 100kHz/(DTR[7:0]+1)

Note: The XO_SEL[2:0] bits and the XODIV2 bit in the RF_XO2 register should first be set to get the correct reference clock.

RF_CP3 Register – RF CP Control Register 3

Bit	7	6	5	4	3	2	1	0
Name	DLY_SYN2	DLY_SYN1	DLY_SYN0	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	0	0	1	0	1	0

Bit 7~5 **DLY_SYN2~DLY_SYN0**: Synthesiser delay get ready time

000: 16μs
001: 20μs
010: 24μs
011: 28μs
100: 32μs
101: 36μs
110: 40μs
111: 100μs

Bit 4~3 **D4~D0**: Reserved bits – must remain at POR setting

These bits are reserved and must not be changed by the user.

RF_OD1 Register – RF MMD and OD Control Register 1

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	1	0	0

Bit 7~0 **D7~D0**: Multi-mode divider and output division

This register is used to control the multi-mode divider and output division in the RF circuitry. Note that different setting values should be written into this register for different RF frequency band applications. The recommended setting values are summarised in the following.

RF Frequency Band	315MHz	433MHz	868/915MHz
RF_OD1 Setting Values	08H	04H	00H

RF_OD3 Register – RF MMD and OD Control Register 3

Bit	7	6	5	4	3	2	1	0
Name	—	—	DLY_SXPD1	DLY_SXPD0	—	D2	D1	D0
R/W	—	—	R/W	R/W	—	R/W	R/W	R/W
POR	—	—	0	1	—	1	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **DLY_SXPD1~DLY_SXPD0**: Delay time for SX_EN (1→0)
00: 25μs
01: 50μs
10: 75μs
11: 100μs

Bit 3 Unimplemented, read as "0"

Bit 2~0 **D2~D0**: Reserved bits – must remain at POR setting.
These bits are reserved and must not be changed by the user.

RF_VCO1 Register – RF VCO Control Register 1

Bit	7	6	5	4	3	2	1	0
Name	VCO_SWHB	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	1	0	0	0	0

Bit 7 **VCO_SWHB**: VCO 2.5GHz frequency band switch control
0: Other frequency bands
1: 315MHz Band
This bit is used to select the VCO 315MHz frequency band. If this bit is set to 1, the VCO 315MHz is selected. Other VCO frequency bands except 315MHz will be selected when this bit is set to 0.

Bit 6~0 **D6~D0**: Frequency Band control.

Note that different setting values should be written into this register for different RF frequency band applications. The recommended setting values are summarized in the following.

RF Frequency Band	315MHz	433MHz	868/915MHz
RF_VCO1 Setting Values	90H	10H	10H

RF_VCO2 Register – RF VCO Control Register 2

Bit	7	6	5	4	3	2	1	0
Name	DFCSF	—	VCO_DFC4	VCO_DFC3	VCO_DFC2	VCO_DFC1	VCO_DFC0	DFC_OW
R/W	R	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	1	0	0	0	0	0

Bit 7 **DFCSF**: DFC select failure flag – read only
This bit status is updated after the VCO calibration has completed. If the calibration result exceeds the DFC curve design range, this bit will be set high to indicate that no DFC is found in the current selected channel. If the DFC has failed, it means the VCO can not operate at the desired frequency or there may be other abnormalies.

Bit 6 Unimplemented, read as "0"

Bit 5~1 **VCO_DFC4~ VCO_DFC0**: VCO DFC curve trim value
00000: Minimum frequency
↓ : ↓
11111: Maximum frequency
These bits are used to set the various capacitor array values corresponding to different frequency bands to increase the VCO frequency covering range. The larger the field value is set, the higher VCO frequency is obtained. The VCO DFC trim value in this bit field can be determined by the software manual setting or hardware auto calibration fitting selected by the DFC_OW bit.

Bit 0 **DFC_OW**: VCO DFC curve trim value source selection
0: VCO DFC curve trim value is determined by auto calibration result
1: VCO DFC curve trim value is determined by manual setting

This bit is used to select the VCO DFC curve trim value source. When this bit is set to 0, the VCO DFC curve trim value will be derived from the auto calibration result regardless of the VCO_DFC bit field value. However, the VCO DFC curve trim value will be determined by the VCO_DFC bit field value rather than the auto calibration result when this bit is set to 1. When this bit is set to 0, the VCO_DFC bit field will return the auto calibration result for the VCO DFC curve trim value if a VCO_DFC field read operation is executed.

Two examples for implementing a DFC calibration are provided below.

- Implement DFC calibration after each power on condition
  Step 1: System power on reset, DFC_OW=0 and RF_PDB=0 by default option
  Step 2: Set ACAL_EN=1 to enable the auto DFC calibration, and wait until ACAL_EN is cleared to 0
  Step 3: RF enters power down mode because RF_PDB=0
  Step 4: RF is powered on by setting RF_PDB=1 and keep DFC_OW=0
  Step 5: Set ACAL_EN=1 to enable the auto DFC calibration, and wait until ACAL_EN is cleared to 0
- Implement DFC calibration only after a system power on reset
  Step 1: System power on reset, DFC_OW=0 and RF_PDB=0 by default option
  Step 2: RF is powered on by setting RF_PDB=1
  Step 3: Set ACAL_EN=1 to enable the auto DFC calibration, and wait until ACAL_EN is cleared to 0
  Step 4: Backup VCO_DFC[4:0] to RAM
  Step 5: RF enters power down mode by setting RF_PDB=0
  Step 6: RF is powered on by setting RF_PDB=1 and keep DFC_OW=0
  Step 7: Restore VCO_DFC[4:0] from RAM
  Step 8: Set DFC_OW=1

RF_TX2 Register – RF TX Control Register 2

Bit	7	6	5	4	3	2	1	0
Name	CT_PAD3	CT_PAD2	CT_PAD1	CT_PAD0	—	CT_TXLDO1	CT_TXLDO0	D0
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	1	1	0	1	—	0	0	0

Bit 7~4 **CT_PAD3~CT_PAD0**: TX PAD linear power control

CT_PAD[3:0]	Output Power Fine Tune	CT_PAD[3:0]	Output Power Fine Tune
1100	0dBm_L3	0101	10dBm_L1
1000	0dBm_L2	0001	10dBm_L0
0100	0dBm_L1	111x	13dBm_L3
0000	0dBm_L0	101x	13dBm_L2
1101	10dBm_L3	011x	13dBm_L1
1001	10dBm_L2	001x	13dBm_L0

Note: 1. xdBm_L0~xdBm_L3 are used for output power fine tune.
2. Power consumption: xdBm_L0 > xdBm_L1 > xdBm_L2 > xdBm_L3.

Bit 3 Unimplemented, read as "0"

Bit 2~1 **CT_TXLDO**: XO LDO voltage setting
00: 1.35V
01: 1.5V
10: 1.65V
11: 1.8V

Bit 0 **D0**: Reserved bits

RF_DFC_CAL Register – RF VCO DFC Calibration Control Register

Bit	7	6	5	4	3	2	1	0
Name	CT_MMDLDO1	CT_MMDLDO0	—	D4	D3	D2	D1	D0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	0	0	0	0

Bit 7~6 **CT_MMDLDO1~CT_MMDLDO0**: MMD LDO voltage setting
00: 1.35V
01: 1.5V
10: 1.65V
11: 1.8V

Bit 5 Unimplemented, read as "0"

Bit 4~0 **D4~D0**: Reserved bits – must remain at the POR setting
These bits are reserved and must not be changed by the user.

RF_LDO Register – RF LDO Control 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	1	1	0	0	0

Bit 7~0 **D7~D0**: RF synthesizer and VCO LDO function control

This register is used to control the overall LDO functions for the RF synthesizer and VCO circuitry. Note that different setting values should be written into this register for different RF frequency band applications. The recommended setting values are summarised in the following.

RF Frequency Band	315MHz	433MHz	868/915MHz
RF_LDO Setting Values	18H	18H	14H

RF_XO1 Register – RF XO Control Register 1

Bit	7	6	5	4	3	2	1	0
Name	XSHIFT1	XSHIFT0	—	XO_ TRIM4	XO_ TRIM3	XO_ TRIM2	XO_ TRIM1	XO_ TRIM0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	1	0	—	1	0	1	0	1

Bit 7 **XSHIFT[1**:0]: Internal capacitor load coarse shift for the crystal oscillator
Capacitor step = 4.5pF, XSHIFT field value = 0 ~ 3
Total Capacitor load≈7 + XSHIFT[1:0]×4.5 + XO_TRIM[4:0]×0.2, unit: pF

Bit 5 Unimplemented, read as "0"

Bit 4~0 **XO_TRIM[5**:0]: Internal capacitor load trim value for the crystal oscillator
Capacitor step = 0.2pF, XO_TRIM field value = 0 ~ 31
Total Capacitor load≈7 + XSHIFT[1:0]×4.5 + XO_TRIM[4:0]×0.2, unit: pF

RF_XO2 Register – RF XO Control Register 2

Bit	7	6	5	4	3	2	1	0
Name	—	D6	D5	—	XODIV2	XO_SEL2	XO_SEL1	XO_SEL0
R/W	—	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	—	0	1	—	0	0	1	1

Bit7 Unimplemented, read as "0"

Bit 6~5 **D6~D5**: Reserved bits – must remain at the POR setting
These bits are reserved and must not be changed by the user.

Bit 4 Unimplemented, read as "0"

Bit 3 **XODIV2**: XO output divided by 2 selection
0: Without any division
1: Divided by 2
It is recommended to clear this bit to zero.

Bit 2~0 **XO_SEL2~XO_SEL0**: XO output selection
000: 12MHz
001: 12.8MHz
010: Reserved
011: 16MHz
100: 19.2MHz
101/110/111: Reserved

The external crystal should be properly selected using the XODIV2 and XO_SEL bit field. Different setting values should be written into this register for different external crystal selections and RF frequency band applications. The recommended setting values are summarised in the following.

RF Frequency Band / External Crystal Selection	315MHz	433MHz	868/915MHz
12MHz	0010 0000	0010 0000	0100 0000
12.8MHz	0010 0001	0010 0001	0100 0001
16MHz	0010 0011	0010 x011	0100 x011
19.2MHz	0010 0100	0010 x100	0100 x100

"x": 0 or 1

Modulation Modes and Operating Modes Selection

- Modulation Modes
  There are two RF modulation modes for the devices, FSK mode and OOK mode, which are selected by the FSK_EN bit.
  In the OOK modulation mode, the RFOUT pin outputs the RF carry signal with a frequency f_C that is selected by channel code DN[6:0] and DK[19:0]. The RF carry signal on/off is controlled by transmitting data at the required data rate.
  In the FSK modulation mode, the RFOUT pin outputs the RF signal with (f_C+f_{DEV}) for data "1" and (f_C-f_{DEV}) for data "0". The RF carry signal on/off is controlled by transmitting data at the determined data rate.
- Operating Modes
  There are two RF operating modes for the devices, Burst Mode and Direct Mode, which are selected by the DIR_EN bit.
  In the Burst Mode, the data to be transmitted is from the FIFO and the RF block is controlled by a state machine. Users just need to write data to the FIFO and set the TX_STROBE bit high to start the transmission and wait for its completion. This is an easy mode to use.
  In the Direct Mode, the data to be transmitted is configured by the DTXD bit in the RF_FIFO_CTRL4 register. The on/off timing for the RF functional block is controlled by specific program squences as described below. This mode provides the maximum flexibility but requires users to take more attention to the block control and data to be transmitted.

RF_OPER Register – RF Operation Control Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	FSK_EN	DIR_EN	—	—	—	TX_STROBE
R/W	—	—	R/W	R/W	—	—	—	R/W
POR	—	—	0	0	—	—	—	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **FSK_EN**: RF output modulation mode selection
0: OOK Mode, PAD enable is controlled by TXD directly
1: FSK Mode, PAD enable is forced to 1 and TXD modulates the TX frequency

Bit 4 **DIR_EN**: RF output operating mode selection
0: Burst Mode, TX data is read from FIFO, RF is controlled by state machine (if FIFO transmission is complete, RF TX will enter the standby mode)
1: Direct Mode, TX data comes from DTXD bit in the RF_FIFO_CTRL4 register directly and RF state is controlled by SX_EN (first enable) and TX_EN (second enable)

Bit 3~1 Unimplemented, read as "0"

Bit 0 **TX_STROBE**: TX strobe to start the burst mode data transfer from RF
0: TX packet has been sent completely
1: Initial a TX transmission
Setting this bit high will initial the TX transmission procedure automatically, in which condition, all register settings are not allowed to change. This bit will be cleared to 0 automatically when a TX packet has been sent completely, in which condition, a BMTCF interrupt request will be generated. Users should start the next transmission after this bit returns to 0. Users can force the TX transmission to teminate by writting a 0 value to this bit.

RF_MOD1 Register – RF Modulator Control Register 1

Bit	7	6	5	4	3	2	1	0
Name	FDEV7	FDEV6	FDEV5	FDEV4	FDEV3	FDEV2	FDEV1	FDEV0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	1	0	0	0	0

Bit 7~0 **FDEV7~FDEV0**: Set the frequency deviation in FSK mode
These bits together with the FDEV10~FDEV9 bits in the RF_MOD2 register set the frequency deviation in FSK mode.

RF_MOD2 Register – RF Modulator Control Register 2

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	FDEV10	FDEV9	FDEV8
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	1

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **FDEV10~FDEV8**: Set the frequency deviation in FSK mode
These bits together with the FDEV7~FDEV0 bits in the RF_MOD1 register set the frequency deviation in FSK mode.
FDEV[10:0]=DEC2HEX (INT($(2^{17}-1)/f_{XTAL}$)×f_D))
For example, if f_{XTAL}=16MHz, XODIV2=0, f_D=50kHz, then FDEV[10:0]=19BH.

RF_OPMOD Register – RF Operation Mode Control Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	ACAL_EN	TX_EN	SX_EN
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **ACAL_EN**: Auto calibration enable control
0: Disable
1: Enable
This bit can be set or reset by the application program to activate or terminate the auto calibration function. It will automatically be cleared to 0 by hardware when the auto calibration is completed.

Bit 1 **TX_EN**: Transmitter control (for Direct Mode only)
0: Disable
1: Enable
This bit is only used in the Direct Mode, it is not used in the Burst Mode.

Bit 0 **SX_EN**: Synthesizer (PLL On) control (for Direct Mode only)
0: Disable
1: Enable
This bit is only used in the Direct Mode, it is not used in the Burst Mode.

TX FIFO Mode in Burst Mode

In the Burst mode, the data to be transmitted comes from the FIFO into which data is pre-written by the MCU. There are three FIFO modes to support various applications. These are Simple FIFO mode, Block FIFO mode and Extend FIFO mode. To use the FIFO in the burst mode, set the RST_TX_FF bit in the RF_FIFO_CTRL3 register to 1 to reset the FIFO pointer and buffer. After this, the FIFO is in the initial state same as power on reset.

- Simple FIFO Mode
  This FIFO mode is used in general applications. The data length should not exceed 128 bytes. To use the simple FIFO mode, the MCU must write the data to be transmitted to the FIFO by accessing the RF_FIFO_CTRL1 register (FFDATA[7:0]). The transmit sequence to the transmitter is first written byte first out and is MSB first out in each byte. Users should first determine the transmit data packet format such as the preamble, ID code and packet encoding such as the FEC (Forward Error Correction), CRC (Cyclic Redundancy Check), whiting and etc. After the FIFO has been filled completely, set the FFSA[6:0] bits in the RF_FIFO_CTRL3 register to 0 and configure the RF_FIFO_CTRL2 register FFLEN[7:0] to the desire transmition length in bytes. Then configure the FSK_EN, DIR_EN and TX_STROBE bits to start the transmission. After the current transmition is complete, the data will be kept in the FIFO to wait for next transmission.
- Block FIFO Mode
  The Bock FIFO mode is used to support multi-key code applications. Users should write all the key code to the FIFO beforehand. When a key is pressed, the MCU will detect the key, set the RF_FIFO_CTRL3 register FFSA[6:0] to the start address of the target key code, and then set the RF_FIFO_CTRL2 register FFLEN[7:0] to indicate the key code length and finally set the TX_STROBE bit in the RF_OPER register to start the transmission. The maximum FIFO length is also limited to 128 bytes.

- Extend FIFO Mode

  The Extend FIFO mode is used for large data packet length up to 256 bytes. The physical FIFO length is 128 bytes. To extend the available transmit length in one packet, a handshake mechanism is needed between the MCU and FIFO.

  Set the FFMG[1:0] bits in the RF_FIFO_CTRL4 register to determine the FIFO data length margin and set the FFMG_EN bit to enable the margin detect function to remind the MCU. The MCU should write data to FIFO as soon as possible when receiving this remind signal, to avoid the transmission being terminated by FIFO data length low to zero.

RF_FIFO_CTRL1 Register – RF FIFO Control Register 1

Bit	7	6	5	4	3	2	1	0
Name	FFDATA7	FFDATA6	FFDATA5	FFDATA4	FFDATA3	FFDATA2	FFDATA1	FFDATA0
R/W	W	W	W	W	W	W	W	W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **FFDATA7~FFDATA0**: FIFO data port

Writing data to this port will write data to the FIFO. Reading data from this port will read data from the top of the FIFO.

RF_FIFO_CTRL2 Register – RF FIFO Control Register 2

Bit	7	6	5	4	3	2	1	0
Name	FFLEN7	FFLEN6	FFLEN5	FFLEN4	FFLEN3	FFLEN2	FFLEN1	FFLEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	1	1	1	1	1

Bit 7~0 **FFLEN7~FFLEN0**: TX data length (used for Burst mode only)

The number of data bytes to be transmitted is FFLEN[7:0]+1. In the Simple FIFO mode and Block FIFO mode, the FFLEN[7:0] is limited to 7Fh. Users should not set this register to be greater than this value.

RF_FIFO_CTRL3 Register – RF FIFO Control Register 3

Bit	7	6	5	4	3	2	1	0
Name	RST_TX_FF	FFSA6	FFSA5	FFSA4	FFSA3	FFSA2	FFSA1	FFSA0
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 **RST_TX_FF**: Reset TX FIFO

0: Do not reset TXFIFO to its initial state or the reset action is completed

1: Reset TX FIFO to initial state

Setting this bit high will reset the TX FIFO to its initial state. After the reset action is complete, this bit will be automatically cleared.

Bit 6~0 **FFSA7~FFSA0**: FIFO start address to send out data – used for the Block FIFO mode

RF_FIFO_CTRL4 Register – RF FIFO Control Register 4

Bit	7	6	5	4	3	2	1	0
Name	DTXD	—	—	—	TXFFLT	FFMG_EN	FFMG1	FFMG0
R/W	R/W	—	—	—	R	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	1

Bit 7 **DTXD**: Direct mode TX data setting

Bit 6~4 Unimplemented, read as "0"

Bit 3 **TXFFLT**: TX FIFO length less than the threshold length specified by FFMG[1:0] (read only)
0: TX FIFO length is greater than the threshold length specified by FFMG[1:0]
1: TX FIFO length is less than or equal to threshold length specified by FFMG[1:0]
If the FFMG_EN bit equals to 0, the TXFFLT bit always keeps at 0. If the FFMG_EN bit equals to 1, the TXFFLT bit indicates the TX FIFO length status. When this bit is set high by hardware, it indicates that the TX FIFO length is less than or equal to the threshold length specified by FFMG[1:0], in which case, an FFMGF interrupt request will be generated.

Bit 2 **FFMG_EN**: TX FIFO length margin detect enable
0: Disable
1: Enable

Bit 1~0 **FFMG1~FFMG0**: TX FIFO length margin - indicates the remained number of data bytes in the FIFO
00: 4 bytes
01: 8 bytes
10: 16 bytes
11: 32 bytes

RF Channel Setup

The RF channel setup is implemented using four registers, the RF_SX1 ~ RF_SX4 registers.

RF_SX1 Register – RF Fractional-N Synthesizer Control Register 1

Bit	7	6	5	4	3	2	1	0
Name	—	DN6	DN5	DN4	DN3	DN2	DN1	DN0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	1	1	0	1	1	0

Bit 7 Unimplemented, read as "0"

Bit 6~0 **DN6~DN0**: Integer of dividend for MMD, supports 32 to 127
Since the MMD data is generated from a delta-sigma modulator, the DN support range will be affected. The real range will be (32+3) ~ (127-4), i.e. 35~123.
Set initial value to XO=16MHz and TX band=433.92MHz:
For example, if Crystal=XO=16MHz, XODIV2=0, and RF_OD1=04H;
VCO frequency=TX frequency×4=433.92MHz×4=1735.68MHz;
$XO\times(DN+DK/2^{20})=16MHz\times(DN+DK/2^{20})=VCO\ frequency/2=1735.68Mhz/2=867.84MHz$;
$(DN+DK/2^{20})=54.24$, so DN[6:0]=36H=54, DK[19:0]=3D70AH=251658.

RF_SX2 Register – RF Fractional-N Synthesizer Control Register 2

Bit	7	6	5	4	3	2	1	0
Name	DK7	DK6	DK5	DK4	DK3	DK2	DK1	DK0
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	1	0

Bit 7~0 **DK7~DK0**: Low byte of 20-bit fractional of dividend for MMD
Set an initial value to implement XO=16MHz and TX band=433.92MHz.

RF_SX3 Register – RF Fractional-N Synthesizer Control Register 3

Bit	7	6	5	4	3	2	1	0
Name	DK15	DK14	DK13	DK12	DK11	DK10	DK9	DK8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	0	1	0	1	1	1

Bit 7~0 **DK15~DK8**: Middle byte of 20-bit fractional of dividend for MMD
Set an initial value to implement XO=16MHz and TX band=433.92MHz.

RF_SX4 Register – RF Fractional-N Synthesizer Control Register 4

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	DK19	DK18	DK17	DK16
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	1	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **DK19~DK8**: High byte of 20-bit fractional of dividend for MMD
Set an initial value to implement XO=16MHz and TX band=433.92MHz.

RF Channel Setup Description (VCO operating frequency: 1.7~1.9GHz, 2.5GHz)

The XODIV2 bit and the RF_OD1 register must be configured according to the following two rules:

- Whether the the VCO operating frequency is in the specified range.
- The calculated DN and DK values are within the available range of RF_SX1~RF_SX4 registers.

The following are five frequency configuration examples (if Crystal=XO=16MHz):

(1) 315MHz

XODIV2=0, RF_OD1=08H

VCO frequency=TX Frequency×8=315MHz×8=2520MHz

XO×(DN+DK/2^{20}) =16MHz×(DN+DK/2^{20})=VCO frequency/2=2520MHz/2=1260MHz

(DN+DK/2^{20})=78.75 → DN[6:0]=4EH=78, DK[19:0]=C0000H=786432

Note: The VCO_SWHB bit should be set to "1" by the user program for switching the VCO to the 2.5GHz band when the TX frequency is 315MHz.

(2) 433MHz

XODIV2=0, RF_OD1=04H

VCO frequency=TX Frequency×4=433MHz×4=1732MHz

XO×(DN+DK/2^{20})=16MHz×(DN+DK/2^{20})=VCO frequency/2=1732MHz/2=866MHz

(DN+DK/2^{20})=54.125 → DN[6:0]=36H=54, DK[19:0]=20000H=131072

(3) 433.92MHz (default setting)

XODIV2=0, RF_OD1=04H

VCO frequency=TX Frequency×4=433.92MHz×4=1735.68MHz

XO×(DN+DK/2^{20})=16MHz×(DN+DK/2^{20})=VCO frequency/2=1735.68MHz/2=867.84MHz

(DN+DK/2^{20})=54.24 → DN[6:0]=36H=54, DK[19:0]=3D70AH=251658

(4) 868MHz

XODIV2=0, RF_OD1=00H

VCO frequency=TX Frequency×2=868MHz×2=1736MHz

XO×(DN+DK/(220))=16MHz×(DN+DK/(2^{20}))=VCO frequency/2=1736MHz/2=868MHz

(DN+DK/(220))=54.25 → DN[6:0]=36H=54, DK[19:0]=40000H=262144

(5) 915MHz

XODIV2=0, RF_OD1=00H

VCO frequency=TX Frequency×2=915MHz×2=1830MHz

XO×(DN+DK/2^{20})=16MHz×(DN+DK/2^{20})=VCO frequency/2=1830MHz/2=915MHz

(DN+DK/2^{20})=57.1875 → DN[6:0]=39H=57, DK[19:0]=30000H=196608

Software Programming Guide

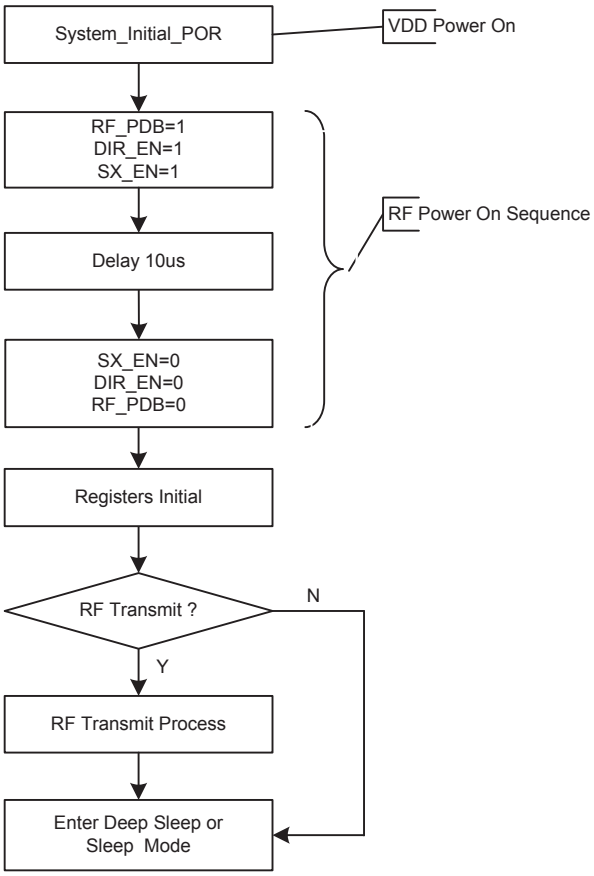
In order to help users to implement the desired transmission, this section provides the reference setup steps for the Burst mode and Direct Mode. The differences between these two modes are the generation of the data to be transmitted and the RF block on/off control, more details are described below.

- Burst Mode
  1. Set the XO_SEL[2:0], XODIV2 and XCLKD2 bits properly according to the selected crystal oscillator.
  2. Set the RF_PDB bit to 1 and the RF circuit will start to operate with the XCLK clock since the XCLK_EN bit has a default value of 1.
  3. If users selected HXT as the system clock source, check the HXTF bit. When this bit is equal to 1 it indicates that the MCU clock is ready. The MCU can use the XCLK_MCU clock as its main clock.
  4. Set the RF related configuration registers.
  5. If it is the first time to power up or the operation environment has large variations, set the ACAL_EN bit in the RF_OPMOD register to start an auto calibration process and poll this bit to wait for the end of the calibration process. Read the value of the VCO_DFC[4:0] bits in the RF_VCO2 register and store it in backup RAM. This action is to save the calibration time when the MCU or RF circuit enters the power down mode. This data will be restored to VCO_DFC[4:0] by a write access to this bit field. Then set the DFC_OW bit in the same register to 1 to force the DFC to select this curve.
  6. Write data to the FIFO. Set FFLEN[7:0] for the desired transmition length in bytes. Set DTR[7:0] for the data rate. Set FDEV[10:0] for the frequency deviation in FSK mode.

7. Set DIR_EN=0 to select the Burst Mode, set FSK_EN=0(OOK)/1(FSK), set TX_STROBE=1 to start the transmission. Then the data in FIFO will be automatically transfered to the RFOUT pin with FSK or OOK modulation.
8. Poll the TX_STROBE bit until it is cleared to 0 by hardware. When the TX_STROBE bit equals to 1, any write accesses to RF configuration registers are not allowed.
9. If users want to resend the previous data, just check the TX_STROBE bit, when it is in a low state set it to 1 again to initial another transmission.
10. If TX_STROBE is in a high state and users want to forcely terminate this transmission, just set the TX_STROBE to low then wait at least 1μs to let the state machine turn off the Power Amplifier Driver. Then users can re-set TX_STROBE to start a new transmission.
11. Set XCLK_EN=0 then clear RF_PDB to low to turn off the RF circuit.

- Direct Mode:
  1. Set the XO_SEL[2:0], XODIV2 and XCLKD2 bits properly according to the selected crystal oscillator.
  2. Set the RF_PDB bit to 1 and the RF circuit will start to operate with the XCLK clock since the XCLK_EN bit has a default value of 1.
  3. If HXT is selected as the system clock source, check the HXTF bit. When this bit is equal to 1 this indicates that the MCU clock is ready. The MCU can use the XCLK_MCU clock as its main clock.
  4. Set the RF related configuration registers.
  5. If it is the first time to power up or the operation environment has large variations, set the ACAL_EN bit in the RF_OPMOD register to start the auto calibration process and poll this bit to wait for the end of the calibration process. Read the value of the VCO_DFC[4:0] bits in the RF_VCO2 register and store it in backup RAM. This action is to save the calibration time when the MCU or RF circuit enters the power down mode. This data will be restored to VCO_DFC[4:0] by a write access to this bit field. Then set the DFC_OW bit in the same register to 1 to force the DFC to select this curve.
  6. Write the DTXD bit with the first data bit to be transmitted. Set FDEV[10:0] for the frequency deviation in the FSK mode.
  7. Set DIR_EN=1 to select the Direct Mode, set FSK_EN=0(OOK)/1(FSK).
  8. Set the SX_EN bit to enable all the synthesizer block functions and implement a delay time to wait for the synthesizer to stabilise.
  9. Set the TX_EN bit to enable the Power Amplifier block function and then start to transmit data via the RFOUT pin.
  10. Continue to update the DTXD bit with a desired data rate until all data transmissions are completed.
  11. Set TX_EN=0 and delay 1μs, then set SX_EN=0.
  12. Set XCLK_EN=0 then clear RF_PDB to low to turn off the RF circuit.

Note: When in the Direct Mode and a data transfer has completed, to allow the RF circuit to enter the power down mode to save power consumption, users must configure the related bits in the correct order as described in step11 and step12, otherwise an undesirable standby current will be generated.



RF Transmitter Process

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, LVD and Timebase, etc.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers falls into three categories. The first is the INTC0~INTC1 registers which setup the primary interrupts, the second is the MFI0~MFI2 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pin	INTnE	INTnF	n=0 or 1
Time Base	TBnE	TBnF	n=0 or 1
Multi-function	MFInE	MFInF	n=0~2
LVD	LVDE	LVDF	—
RF TX FIFO Length Margin Detect	FFMGE	FFMGF	—
RF Burst Mode Transmit Complete	BMTCE	BMTCF	—
CTM	CTMPE	CTMPF	—
	CTMAE	STMAF	
PTM	PTMPE	PTMPF	—
	PTMAE	PTMAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	TB0F	INT1F	INT0F	TB0E	INT1E	INT0E	EMI
INTC1	MFI2F	MFI1F	MFI0F	TB1F	MFI2E	MFI1E	MFI0E	TB1E
MFI0	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE
MFI1	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE
MFI2	—	LVDF	FFMGF	BMTCF	—	LVDE	FFMGE	BMTCE

Interrupt Registers List

INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt edge control for INT0 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	TB0F	INT1F	INT0F	TB0E	INT1E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **INT1F**: INT1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INT0F**: INT0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

Bit 2 **INT1E**: INT1 interrupt control
0: Disable
1: Enable

Bit 1 **INT0E**: INT0 interrupt control
0: Disable
1: Enable

Bit 0 **EMI**: Global interrupt control
0: Disable
1: Enable

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	MFI2F	MFI1F	MFI0F	TB1F	MFI2E	MFI1E	MFI0E	TB1E
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 **MFI2F**: Multi-function interrupt 2 request flag
0: No request
1: Interrupt request

Bit 6 **MFI1F**: Multi-function interrupt 1 request flag
0: No request
1: Interrupt request

Bit 5 **MFI0F**: Multi-function interrupt 0 request flag
0: No request
1: Interrupt request

Bit 4 **TB1F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **MFI2E**: Multi-function interrupt 2 control
0: Disable
1: Enable

Bit 2 **MFI1E**: Multi-function interrupt 1 control
0: Disable
1: Enable

Bit 1 **MFI0E**: Multi-function interrupt 0 control
0: Disable
1: Enable

Bit 0 **TB1E**: Time Base 0 interrupt control
0: Disable
1: Enable

MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **CTMAF**: CTM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTMPF**: CTM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **CTMAE**: CTM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **CTMPE**: CTM Comparator P match interrupt control
0: Disable
1: Enable

MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PTMAF**: PTM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTMPF**: PTM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTMAE**: PTM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTMPE**: PTM Comparator P match interrupt control
0: Disable
1: Enable

MFI2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	LVDF	FFMGF	BMTCF	—	LVDE	FFMGE	BMTCE
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 **LVDF**: LVD interrupt request flag
0: No request
1: Interrupt request

Bit 5 **FFMGF**: RF TX FIFO Length Margin detect interrupt request flag
0: No request
1: Interrupt request

Bit 4 **BMTCF**: RF Burst Mode Transmit Complete interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **LVDE**: LVD interrupt control
0: Disable
1: Enable

Bit 1 **FFMGE**: RF TX FIFO Length Margin detect interrupt control
0: Disable
1: Enable

Bit 0 **BMTCE**: RF Burst Mode Transmit Complete interrupt control
0: Disable
1: Enable

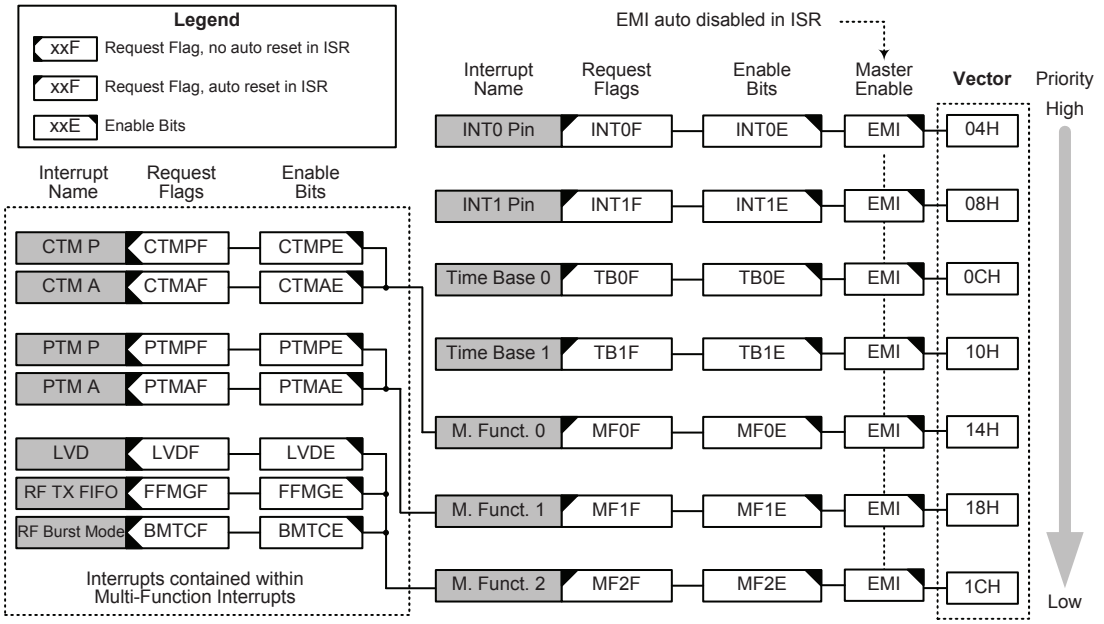
Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A match etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a "JMP" which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a "RETI", which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all 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.



Interrupt Structure

External Interrupts

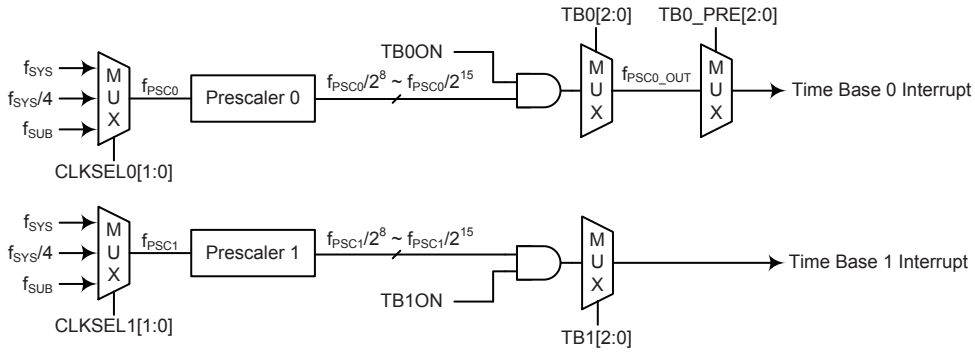
The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The Time Base clock source, f_{PSC0} or f_{PSC1}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the PSCR0 and PSCR1 register respectively.



Time Base Interrupts

PSCR0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL01	CLKSEL00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL01~CLKSEL00**: Prescaler 0 clock source f_{PSC0} selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}
Care must be taken that when the MCU enters the DEEP SLEEP Mode, the Time Base 0 clock source comes from f_{LIRC}, irrespective of the prescaler 0 clock source selection.

PSCR1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL11	CLKSEL10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL11~CLKSEL10**: Prescaler 1 clock source f_{PSC1} selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

TB0C Register

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	TB0_PRE2	TB0_PRE1	TB0_PRE0	—	TB02	TB01	TB00
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 **TB0ON**: Time Base 0 Control
0: Disable
1: Enable

Bit 6~4 **TB0_PRE2~TB0_PRE0**: Select Time Base 0 Time-out Period from f_{PSC0_OUT}
000: $2^0/f_{PSC0\_OUT}$
001: $2^1/f_{PSC0\_OUT}$
010: $2^2/f_{PSC0\_OUT}$
011: $2^3/f_{PSC0\_OUT}$
100: $2^4/f_{PSC0\_OUT}$
101: $2^5/f_{PSC0\_OUT}$
110: $2^6/f_{PSC0\_OUT}$
111: $2^7/f_{PSC0\_OUT}$

Bit 3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Select Time Base 0 Time-out Period from f_{PSC0}
000: $2^8/f_{PSC0}$
001: $2^9/f_{PSC0}$
010: $2^{10}/f_{PSC0}$
011: $2^{11}/f_{PSC0}$
100: $2^{12}/f_{PSC0}$
101: $2^{13}/f_{PSC0}$
110: $2^{14}/f_{PSC0}$
111: $2^{15}/f_{PSC0}$

TB1C Register

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	—	—	—	—	TB12	TB11	TB10
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB1ON**: Time Base 1 Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB12~TB10**: Select Time Base 1 Time-out Period from f_{PSC1}
000: $2^8/f_{PSC1}$
001: $2^9/f_{PSC1}$
010: $2^{10}/f_{PSC1}$
011: $2^{11}/f_{PSC1}$
100: $2^{12}/f_{PSC1}$
101: $2^{13}/f_{PSC1}$
110: $2^{14}/f_{PSC1}$
111: $2^{15}/f_{PSC1}$

Multi-function Interrupts

Within the devices there are up to three Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupts, LVD Interrupt, RF FFMG length margin Interrupt and RF Burst Mode Transmit Complete Interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags, MFInF are set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and 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 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.

RF TX FIFO Length Margin Detect Interrupt

The RF TX FIFO Length Margin Detect interrupt is contained within the Multi-function Interrupt. A TX FIFO Length Margin Detect Interrupt request will take place when its interrupt request flag, FFMGF, is set, which occurs when the TX FIFO length is less than the threshold length. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and TX FIFO Length Margin Detect Interrupt enable bit, FFMGE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and the previously mentioned condition occurs, a subroutine call to the respective interrupt vector will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the FFMGF flag will not be automatically cleared, it has to be cleared by the application program.

RF Burst Mode Transmit Complete Interrupt

The RF Burst Mode Transmit Complete interrupt is contained within the Multi-function Interrupt. A RF Burst Mode Transmit Complete Interrupt request will take place when its interrupt request flag, BMTCF, is set, which occurs when the TX packet has been sent completely in the Burst Mode. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and RF Burst Mode Transmit Complete Interrupt enable bit, BMTCE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and the previously mentioned condition occurs, a subroutine call to the respective interrupt vector will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the BMTCF flag will not be automatically cleared, it has to be cleared by the application program.

LVD Interrupt

The Low Voltage Detector Interrupt is contained within the Multi-function Interrupt. An LVD Interrupt request will take place when the LVD Interrupt request flag, LVDF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVDE, and associated Multi-function interrupt enable bit, 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 Multi-function Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the LVDF flag will not be automatically cleared, it has to be cleared by the application program.

TM Interrupts

The Compact and Periodic Type TMs each have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. For all of the TM types there are two interrupt request flags and two enable control bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFInE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MFInF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins or a low power supply voltage may cause their respective interrupt flag to be set high and consequently generate an interrupt.

The Time Base 0 interrupt has the capacility of waking up the microcontroller when in the DEEP SLEEP Mode, where the Time Base 0 clock source comes from f_{LIRC}.

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 DEEP SLEEP, SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program.

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MFInF, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

It is recommended that programs do not use the “CALL” instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a CALL subroutine is executed in the interrupt subroutine.

Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode and the Time Base 0 interrupt can wake-up the microcontroller when it is in the DEEP SLEEP 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 the SLEEP, IDLE or DEEP SLEEP 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.

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}, and provide a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of eight fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} voltage is above the preset low voltage value. The ENLVD 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	—	—	LVDO	ENLVD	—	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6 Unimplemented, read as “0”

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

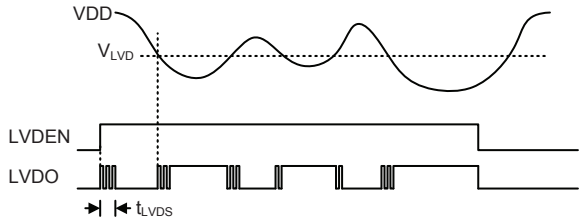
Bit 4 **ENLVD**: Low Voltage Detector Enable control
0: Disable
1: Enable

Bit 3 Unimplemented, read as “0”

Bit 2~0 **VLVD2~VLVD0**: LVD Voltage selection
000: 1.9V
001: 2.0V
010: 2.2V
011: 2.4V
100: 2.8V
101: 2.9V
110: 3.0V
111: 3.6V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.9V and 3.6V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is in the DEEP SLEEP, SLEEP or IDLE mode, the low voltage detector will remain active if the ENLVD bit is 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} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

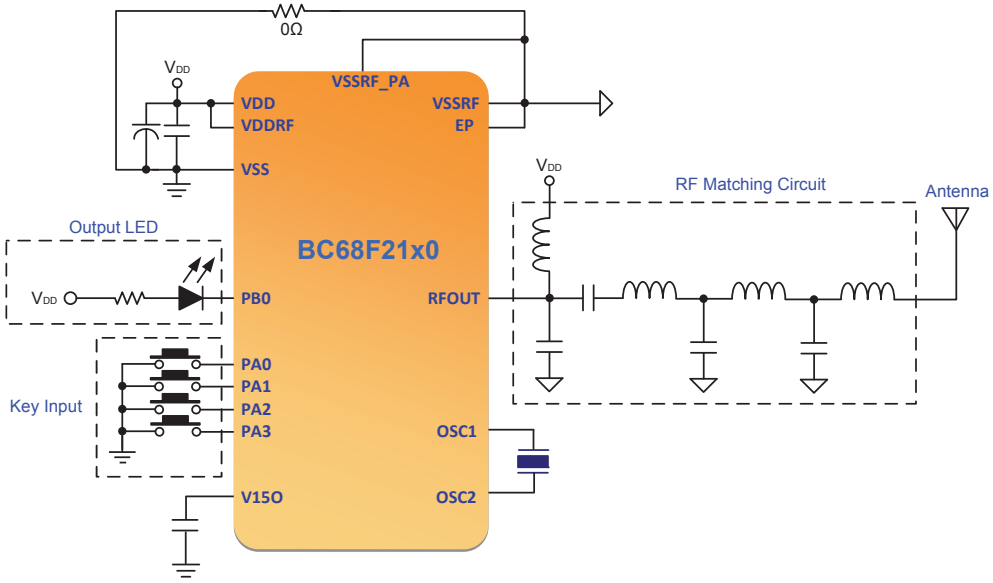
The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. In this case, the LVDF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} falls below the preset LVD voltage. This will cause the device to wake-up from the SLEEP or IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVDF flag should be first set high before the device enters the SLEEP or IDLE Mode. Note that an LVD interrupt request will not cause the device to be woken-up from the DEEP SLEEP Mode, where the device must be woken-up using other methods which have been described in the Wake-up section.

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later using the application program. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Watchdog Timer Option	
1	Watchdog Timer Function: 1. Always enable 2. By WDTC control
LVR Option	
2	LVR Function: 1. Disable 2. Enable

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 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 up to three 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.
3. For the "CLR WDT" instruction the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after the "CLR WDT" instructions is executed. Otherwise the TO and PDF flags remain unchanged.

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 sections except sector 0, the extended instruction can be used to access the data memory instead of using the indirect addressing access to improve the CPU firmware performance.

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 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 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 up to four cycles are required, if no skip takes place two cycles is required.
2. Any extended instruction which changes the contents of the PCL register will also require three cycles for execution.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry

Description: The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + [m] + C

Affected flag(s): OV, Z, AC, C, SC

ADCM A,[m] Add ACC to Data Memory with Carry

Description: The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation: [m] ← ACC + [m] + C

Affected flag(s): OV, Z, AC, C, SC

ADD A,[m] Add Data Memory to ACC

Description: The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + [m]

Affected flag(s): OV, Z, AC, C, SC

ADD A,x Add immediate data to ACC

Description: The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + x

Affected flag(s): OV, Z, AC, C, SC

ADDM A,[m] Add ACC to Data Memory

Description: The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation: [m] ← ACC + [m]

Affected flag(s): OV, Z, AC, C, SC

AND A,[m] Logical AND Data Memory to ACC

Description: Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation: ACC ← ACC "AND" [m]

Affected flag(s): Z

AND A,x Logical AND immediate data to ACC

Description: Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.

Operation: ACC ← ACC "AND" x

Affected flag(s): Z

ANDM A,[m] Logical AND ACC to Data Memory

Description: Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation: [m] ← ACC "AND" [m]

Affected flag(s): Z

CALL addr Subroutine call

Description: Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation: Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s): None

CLR [m] Clear Data Memory

Description: Each bit of the specified Data Memory is cleared to 0.

Operation: [m] ← 00H

Affected flag(s): None

CLR [m].i Clear bit of Data Memory

Description: Bit i of the specified Data Memory is cleared to 0.

Operation: [m].i ← 0

Affected flag(s): None

CLR WDT Clear Watchdog Timer

Description: The TO, PDF flags and the WDT are all cleared.

Operation: WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CPL [m] Complement Data Memory

Description: Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation: [m] ← $\overline{[m]}$

Affected flag(s): Z

CPLA [m] Complement Data Memory with result in ACC

Description: Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation: ACC ← $\overline{[m]}$

Affected flag(s): Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description: Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation: [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s): C

DEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

Description This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

OR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

ORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

Description The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation Program Counter ← Stack

Affected flag(s) None

RET A,x Return from subroutine and load immediate data to ACC

Description The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

Description The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

RLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

RR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory is rotated right by 1 bit with bit 0
rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s): C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: ACC ← 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] - $\overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: [m] ← 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 Data Memory is not 0

Description 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].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

Description 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 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 pair (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 to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

XOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Extended Instruction Definition

The extended instructions are used to directly access the data stored in any data memory sections.

LADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

LADCM A,[m] Add ACC to Data Memory with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

LADD A,[m] Add Data Memory to ACC

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

LADDM A,[m] Add ACC to Data Memory

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

LAND A,[m] Logical AND Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

LANDM A,[m] Logical AND ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″AND″ [m]

Affected flag(s) Z

LCLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

LCLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

LCPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation $[m] \leftarrow \overline{[m]}$

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow \overline{[m]}$

Affected flag(s) Z

LDAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

LDEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation $[m] \leftarrow [m] - 1$

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

LINC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation $[m] \leftarrow [m] + 1$

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] + 1$

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

LMOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

LOR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

LORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

LRL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

LRLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

LRLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

LRLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

LRR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

LRRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s) None

LRRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

LRRCA [m] Rotate Data Memory right through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s) C

LSBC A,[m] Subtract Data Memory from ACC with Carry

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m] − 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 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

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 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

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 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

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 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

LSNZ [m].i Skip if Data Memory is not 0

Description 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].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

Description 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 two 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 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

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 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

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 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

LTABRD [m] Read table (current page) to TBLH and Data Memory

Description The low byte of the program code (current 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

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 to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code 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

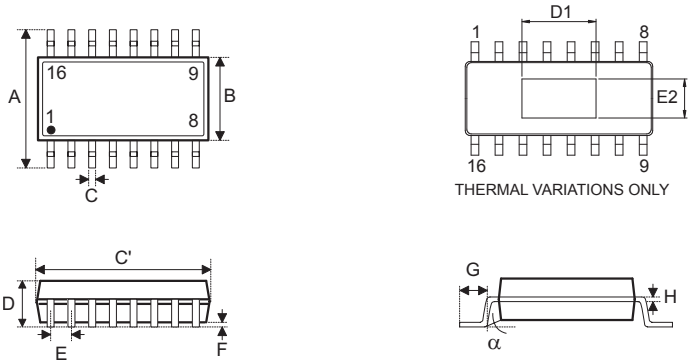
Package Information

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.

- Further Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- Packing Meterials Information
- Carton information

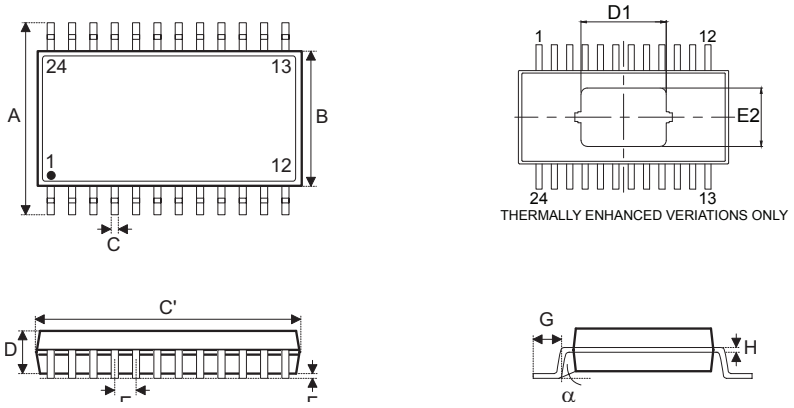
16-pin NSOP (150mil) Outline Dimensions (Exposed Pad)



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C’	—	0.390 BSC	—
D	—	—	0.069
D1	0.059	—	—
E	—	0.050 BSC	—
E2	0.039	—	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.00 BSC	—
B	—	3.90 BSC	—
C	0.31	—	0.51
C’	—	9.90 BSC	—
D	—	—	1.75
D1	1.50	—	—
E	—	1.27 BSC	—
E2	1.00	—	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

24-pin SSOP (150mil) Outline Dimensions (Exposed Pad)



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.341 BSC	—
D	—	—	0.069
D1	—	0.140	—
E	—	0.025 BSC	—
E2	—	0.096	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions mm		
	Min.	Nom.	Max.
A	—	6.00 BSC	—
B	—	3.90 BSC	—
C	0.20	—	0.30
C’	—	8.66 BSC	—
D	—	—	1.75
D1	—	3.56	—
E	—	0.635 BSC	—
E2	—	2.44	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°