Sub-1GHz Transmitter Hopping Code Flash MCU

BC68F3132

Revision: V1.20   Date: February 13, 2023

Table of Contents

Features

CPU Features

- Holtek TinyPower™ Technology
- Operating voltage
  - f_{SYS}=4MHz: 2.2V~3.6V
  - f_{SYS}=8MHz: 2.2V~3.6V
  - f_{SYS}=12MHz: 2.7V~3.6V
- Up to 0.33μs instruction cycle with 12MHz system clock
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 4/8/12MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fast power on structure
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K×15
- Data Memory: 128×8
- True EEPROM Memory: 64×8
- Watchdog Timer function
- 9 bidirectional I/O lines
- Programmable I/O port source and sink current as well as slew rate
- Single external interrupt line shared with I/O pin
- Multiple Timer Modules for time measurement, compare match output or PWM output function
- Dual Time-Base functions for generation of fixed time interrupt signals
- Hopping code engine for generating encryption and decryption functions
- Low voltage reset function
- Low voltage detect function
- Package types: 16-pin NSOP-EP/QFN

RF Transmitter Features

- Operating voltage: V_{DD}=2.2V~3.6V
- Frequency bands: 315MHz, 433MHz, 868MHz, 915MHz
- Output Power up to 13dBm
- TX current consumption @ 433MHz
  - 17mA typical (FSK, 10dBm)
  - 11mA typical (OOK, 10dBm, 50% duty cycle)

- OOK symbol rate from 0.5Ksps to 25Ksps, FSK data rate from 0.5Kbps to 50Kbps
- 4-step programmable TX Power: 0dBm, 5dBm, 10dBm, 13dBm
- Fully integrated VCO, on chip loop filter and Fractional-N PLL synthesizer
- RF external 16MHz crystal
- FCC/ETSI Compliant

Development Tools

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

https://www.holtek.com/rf-workshop

General Description

The device is a Flash Memory I/O type 8-bit high performance RISC architecture microcontroller with a fully integrated RF transmitter and a hopping code engine, specifically designed for wireless applications.

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

With regard to internal timers, the device includes multiple extremely flexible Timer Modules, which 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.

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

The integrated RF module is a Sub-1GHz OOK/FSK transmitter for wireless applications in the 315MHz, 433MHz, 470MHz, 868MHz and 915MHz frequency bands. It is a highly integrated and cost effective solution for one-way transmitters. It only needs a crystal, a few external capacitors and a few Power Amplifier (PA) output matching components on PCB to form a complete RF solution. The RF module consists of a highly integrated fractional-N Synthesizer and a Class-E Power Amplifier (PA). As it adopts a fractional-N synthesizer, the users can potentially design their transmitters to operate at a wider frequency range. A class-E Power Amplifier (PA) can deliver up to +13dBm output power. The RF transmitter can support OOK and FSK modulation with symbol rate of up to 25Ksps and data rate of up to 50Kbps, respectively.

The hopping code engine specific to wireless applications is also fully integrated within the device, which is used to generate encryption and decryption functions, implementing secure wireless control. The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the device will be highly capable of providing a cost effective Sub-1GHz RF transmitter solution for a wide range of wireless applications, such as Automotive Remote Keyless Entry (RKE) systems, Automotive immobilizers, Community gates and garage door openers, Identity tokens with usage counters, Burglar alarm systems, Building access, etc.

Block Diagram
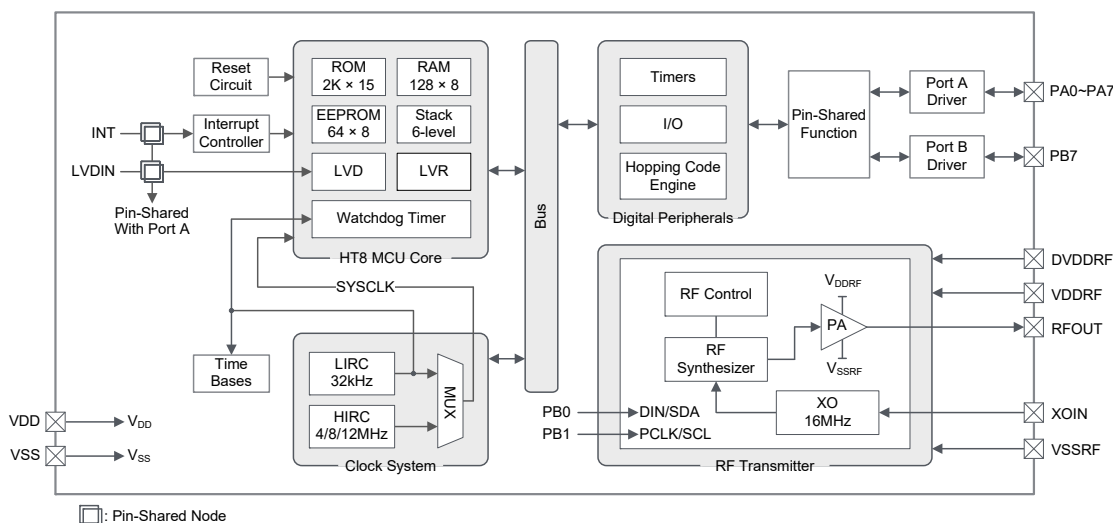
Pin Assignment
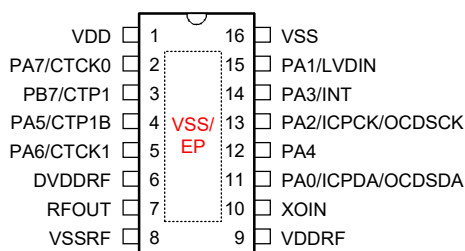
BC68F3132/BC68V3132
16 NSOP-EP-A
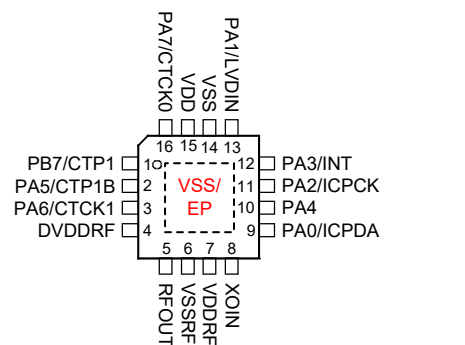
BC68F3132
16 QFN-A

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

2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available
for the BC68V3132 device which is the OCDS EV chip for the BC68F3132 device.

Pin Description

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

Pin Name	Function	OPT	I/T	O/T	Description
PA0/ICPDA/OCDSDA	PA0	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	ICPDA	—	ST	CMOS	ICP data/address pin
	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
PA1/LVDIN	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	LVDIN	PAS0	AN	—	Low voltage detect external input
PA2/ICPCK/OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/INT	PA3	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT	INTEG INTC0	ST	—	External interrupt input
PA4	PA4	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
PA5/CTP1B	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP1B	PAS1	—	CMOS	CTM1 inverted output
PA6/CTCK1	PA6	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK1	—	ST	—	CTM1 clock input
PA7/CTCK0	PA7	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK0	—	ST	—	CTM0 clock input
PB7/CTP1	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP1	PBS1	—	CMOS	CTM1 output
RFOUT	RFOUT	—	—	AN	RF power amplifier output, connected to matching circuit
XOIN	XOIN	—	AN	—	RF external crystal input
DVDDRF	DVDDRF	—	PWR	—	RF digital positive power supply
VDDRF	VDDRF	—	PWR	—	RF positive power supply
VSSRF	VSSRF	—	PWR	—	RF negative power supply
VDD	VDD	—	PWR	—	Digital positive power supply
VSS	VSS	—	PWR	—	Digital negative power supply
VSS/EP	VSS	—	PWR	—	Exposed pad, must be connected to ground

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.

Interconnection Signal Description

Several signals are not connected to external package pins. These signals are interconnection lines between the MCU and the RF transmitter and are listed in the following table.

MCU Signal Name	RF Transmitter Signal Name	Function	Description
PB0	DIN/SDA	PB0	General purpose I/O Internally connected to the RF transmitter input DIN/SDA
		DIN	RF transmitter data input Internally connected to the MCU PB0 line
		SDA	I²C data input Internally connected to the MCU PB0 line
PB1	PCLK/SCL	PB1	General purpose I/O Internally connected to the RF transmitter input PCLK/SCL
		PCLK	RF transmitter clock input Internally connected to the MCU PB1 line
		SCL	I²C clock input Internally connected to the MCU PB1 line

Note: The internal signals, PB0 and PB1, are internally connected to the RF transmitter inputs, DIN/SDA and PCLK/SCL, respectively which should be properly configured to control the RF transmitter. Refer to the "Input/Output Ports" and "RF Transmitter" chapters for more details.

Absolute Maximum Ratings

Supply Voltage ....V_{SS}-0.3V to 3.6V

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

Storage Temperature.. -60°C to 150°C

Operating Temperature.. -40°C to 85°C

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

I_{OL} Total 80mA

Total Power Dissipation 500mW

ESD HBM ....±2kV

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of the device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability. This device is ESD sensitive. HBM (Human Body Mode) is based on the MIL-STD-883.

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	$f_{SYS}=f_{HIRC}=4MHz$	2.2	—	3.6	V
		$f_{SYS}=f_{HIRC}=8MHz$	2.2	—	3.6	
		$f_{SYS}=f_{HIRC}=12MHz$	2.7	—	3.6	
	Operating Voltage – LIRC	$f_{SYS}=f_{LIRC}=32kHz$	2.2	—	3.6	

Operating Current Characteristics

Ta=-40°C~85°C

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

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

Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.2V	WDT off	—	0.11	0.15	2.00	µA
		3V		—	0.11	0.15	2.00	
		2.2V	WDT on	—	1.2	2.4	2.9	
		3V		—	1.5	3.0	3.6	
	IDLE0 Mode – LIRC	2.2V	f_{SUB} on	—	2.4	4.0	4.8	µA
		3V		—	3	5	6	
	IDLE1 Mode – HIRC	2.2V	f_{SUB} on, $f_{SYS}=4MHz$	—	144	200	240	µA
		3V		—	180	250	300	
		2.2V	f_{SUB} on, $f_{SYS}=8MHz$	—	288	400	480	
		3V		—	360	500	600	
		2.7V	f_{SUB} on, $f_{SYS}=12MHz$	—	432	600	720	
		3V		—	540	750	900	

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

A.C. Characteristics

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

High Speed Internal Oscillator – HIRC – Frequency Accuracy

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	4MHz Writer Trimmed HIRC Frequency	3V	25°C	-1%	4	+1%	MHz
			-40°C~85°C	-2%	4	+2%	
		2.2V~3.6V	25°C	-2.5%	4	+2.5%	
			-40°C~85°C	-3%	4	+3%	
	8MHz Writer Trimmed HIRC Frequency	3V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.2V~3.6V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	
	12MHz Writer Trimmed HIRC Frequency	3V	25°C	-1%	12	+1%	MHz
			-40°C~85°C	-2%	12	+2%	
		2.7V~3.6V	25°C	-2.5%	12	+2.5%	
			-40°C~85°C	-3%	12	+3%	

Note: 1. The 3V values for V_{DD} are provided as this is the fixed voltage at which the HIRC frequency is trimmed by the writer.

2. The row below the 3V trim voltage row is provided to show the values for the full V_{DD} range operating voltage.

Low Speed Internal Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	2.2V~3.6V	-40°C~85°C	-8%	32	+8%	kHz
t_{START}	LIRC Start Up Time	—	-40°C~85°C	—	—	100	µs

Operating Frequency Characteristic Curves

System Start Up Time Characteristics

Ta=-40°C~85°C

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

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

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

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

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

Input/Output Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports or Input Pins	—	—	0	—	$0.2V_{DD}$	V
V_{IH}	Input High Voltage for I/O Ports or Input Pins	—	—	$0.8V_{DD}$	—	V_{DD}	V
I_{OL}	Sink Current for I/O Ports	3V	$V_{OL}=0.1V_{DD}$, DRVCC[m]=0 (m=0~3)	2.5	4.0	—	mA
		3V	$V_{OL}=0.1V_{DD}$, DRVCC[m]=1 (m=0~3)	9.5	16.0	—	
I_{OH}	Source Current for I/O Ports	3V	$V_{OH}=0.9V_{DD}$, DRVCC[m]=0 (m=0~3)	-1.5	-2.5	—	mA
		3V	$V_{OH}=0.9V_{DD}$, DRVCC[m]=1 (m=0~3)	-3	-5	—	
R_{PH}	Pull-high Resistance for I/O Ports [Note]	3V	LVPU=0, PxPU=FFH (Px=PA, PB)	20	60	100	kΩ
		3V	LVPU=1, PxPU=FFH (Px=PA, PB)	6.67	15.00	23.00	
I_{LEAK}	Input Leakage Current	3V	$V_{IN}=V_{DD}$ or V_{SS}	—	—	±1	µA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
SR_{RISE}	Output Rising Edge Slew Rate for I/O Ports	3V	SLEWC[m+1:m]=00B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=0 (m=0~3)	85	150	315	V/µs
			SLEWC[m+1:m]=00B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=1 (m=0~3)	95	160	310	
			SLEWC[m+1:m]=01B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=0 (m=0~3)	45	79	180	
			SLEWC[m+1:m]=01B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=1 (m=0~3)	40	68	140	
			SLEWC[m+1:m]=10B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=0 (m=0~3)	20	38	90	
			SLEWC[m+1:m]=10B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=1 (m=0~3)	15	29	65	
			SLEWC[m+1:m]=11B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=0 (m=0~3)	14	24	55	
			SLEWC[m+1:m]=11B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=1 (m=0~3)	10	17	40	
SR_{FALL}	Output Falling Edge Slew Rate for I/O Ports	3V	SLEWC[m+1:m]=00B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=0 (m=0~3)	88	175	340	V/µs
			SLEWC[m+1:m]=00B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=1 (m=0~3)	69	150	310	
			SLEWC[m+1:m]=01B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=0 (m=0~3)	55	100	200	
			SLEWC[m+1:m]=01B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=1 (m=0~3)	40	67	125	
			SLEWC[m+1:m]=10B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=0 (m=0~3)	25	42	85	
			SLEWC[m+1:m]=10B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=1 (m=0~3)	12	21	40	
			SLEWC[m+1:m]=11B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=0 (m=0~3)	17	27	55	
			SLEWC[m+1:m]=11B (m=0, 2, 4, 6), $0.1V_{DD}$ to $0.9V_{DD}$, C_{LOAD}=20pF, DRVCC[m]=1 (m=0~3)	8	12	25	
t_{TCK}	TM Clock Input Pin Minimum Pulse Width	—	—	0.3	—	—	µs
t_{INT}	External Interrupt Minimum Pulse Width	—	—	0.3	—	—	µs

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

Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{RW}	V_{DD} for Read / Write	—	—	V_{DDmin}	—	V_{DDmax}	V
Flash Program Memory / Data EEPROM Memory							
t_{DEW}	Erase / Write Cycle Time – Flash Program Memory	—	—	—	2	3	ms
	Write Cycle Time – Data EEPROM Memory	—	—	—	4	6	ms
I_{DDPGM}	Programming / Erase Current on V_{DD}	—	—	—	—	5.0	mA
E_P	Cell Endurance – Flash Program Memory	—	—	10K	—	—	E/W
	Cell Endurance – Data EEPROM Memory	—	—	100K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

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

LVR/LVD Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 1.7V	-5%	1.7	+5%	V
V_{LVD}	Low Voltage Detection Voltage	≥2.2V	LVD enable, voltage select LVDIN pin=1.25V	-5%	1.25	+5%	V
		—	LVD enable, voltage select 2.0V		2.0		
		—	LVD enable, voltage select 2.2V		2.2		
		—	LVD enable, voltage select 2.4V		2.4		
		—	LVD enable, voltage select 2.7V		2.7		
		—	LVD enable, voltage select 3.0V		3.0		
		—	LVD enable, voltage select 3.3V		3.3		
		—	LVD enable, voltage select 3.6V		3.6		
$I_{LVRLVDBG}$	Operating Current	3V	LVD enable, LVR enable, V_{LVR}=1.7V, V_{LVD}=2.0V	—	—	40	µA
t_{LVDS}	LVDO Stable Time	—	For LVR enable, LVD off → on	—	—	18	µs
		—	For LVR disable, LVD off → on	—	—	150	
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	µs
t_{LVD}	Minimum Low voltage Width to Interrupt	—	—	60	120	240	µs
I_{LVR}	Additional Current for LVR Enable	—	LVD disable	—	—	25	µA
I_{LVD}	Additional Current for LVD Enable	—	LVR disable	—	—	25	µA

Note: If V_{LVD}=1.25V, it is used to detect the LVDIN pin input voltage. Other V_{LVD} choices are used to detect the power supply V_{DD}.

Hopping Code Engine Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
$t_{HP1CYCLE}$	Hopping Code Engine Run 1 LOOPCNT Time	—	—	—	1	2	t_{SYS}

RF Transmitter Electrical Characteristics

Ta=25°C, V_{DD}=3.3V, f_{XTAL}=16MHz, OOK/FSK modulation with Matching circuit
RFOUT is powered by V_{DD}=3.3V, unless otherwise specified

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_{OP}	Operating Temperature	—	-40	—	85	°C
V_{DD}	Supply Voltage	—	2.2	3.3	3.6	V
Current Consumption						
I_{Sleep}	Deep Sleep Mode Current Consumption	—	—	—	0.4	µA
$I_{Standby}$	Idle Mode Current Consumption	XTAL on, Power Amplifier off, Synthesizer on	—	6.5	—	mA
I_{TX}	High Data Current Consumption @ 315MHz (Data=1)	P_{RF}=0dBm	—	11	—	mA
		P_{RF}=10dBm	—	19	—	
		P_{RF}=13dBm	—	24	—	
	High Data Current Consumption @ 433MHz (Data=1)	P_{RF}=0dBm	—	11	—	mA
		P_{RF}=10dBm	—	17	—	
		P_{RF}=13dBm	—	24	—	
I_{TX}	High Data Current Consumption @ 868MHz (Data=1)	P_{RF}=0dBm	—	11	—	mA
		P_{RF}=10dBm	—	19	—	
		P_{RF}=13dBm	—	24	—	
	High Data Current Consumption @ 915MHz (Data=1)	P_{RF}=0dBm	—	12	—	mA
		P_{RF}=10dBm	—	20	—	
		P_{RF}=13dBm	—	25	—	
Pull-high Resistance						
R_{PH}	Pull-high Resistance for I/O Ports	3.3V	—	33	—	kΩ
Transmitter Characteristics						
f_{RF}	RF Frequency Band	315MHz band	—	315	—	MHz
		433MHz band	—	433.92	—	
		868MHz band	—	868.35	—	
		915MHz band	—	915	—	
SR	Symbol Rate	OOK modulation	0.5	—	25	Ksps
DR	Data Rate	FSK modulation (@ f_{DEV}=12.5kHz)	0.5	—	50	Kbps
P_{RF}	RF Transmit Output Power	433MHz band	0	—	13	dBm
		868MHz band	0	—	13	
t_{ST}	RF Transmit Settling Time	Standby mode to Transmit mode	—	370	—	µs
ER_{OOK}	OOK Extinction Ratio	OOK modulation depth	—	70	—	dB
f_{DEV}	Frequency Deviation	FSK modulation @ f_{XTAL}=16MHz	2	—	100	kHz
	Output Blanking	From Deep Sleep to Transmit mode	—	—	1	ms
	One Shot Delay Time	OOK/FSK	4	—	32	ms

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
S.E.$_{TX}$	TX Spurious Emission (P_{RF}=10dBm)	f<1GHz	—	—	-36	dBm
		47MHz<f<74MHz	—	—	-54	
		87.5MHz<f<118MHz				
		174MHz<f<230MHz				
		470MHz<f<862MHz				
		2nd, 3rd Harmonic	—	—	-30	
LO Characteristics						
f$_{LO}$	RF Frequency Coverage Range	315MHz band	290	—	335	MHz
		433MHz band	415	—	490	
		868MHz band	830	—	960	
f$_{STEP}$	LO Frequency Resolution	—	—	—	1	kHz
PN$_{LO}$	433MHz Phase Noise	@ 100kHz offset	—	-76	—	dBc/Hz
		@ 1MHz offset	—	-104	—	
	868MHz Phase Noise	@ 100kHz offset	—	-70	—	
		@ 1MHz offset	—	-100	—	
Crystal Oscillator						
f$_{XTAL}$	XTAL Frequency	—	—	16	—	MHz
ESR	XTAL Equivalent Series Resistance	—	—	—	100	Ω
C$_{LOAD}$	XTAL Capacitor Load	—	—	16	—	pF
TOL $^{(Note)}$	XTAL Tolerance	—	-20	—	+20	ppm
t$_{SU}$	XTAL Startup Time	49US	—	—	1	ms
		3225SMD	—	3	—	

Note: This is the total tolerance including Initial tolerance, Crystal loading, Aging and Temperature dependence. The acceptable crystal tolerance depends on RF frequency and channel spacing/band width.

I²C Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
f$_{SCL}$	Serial Clock Frequency	—	—	—	1	MHz
t$_{BUF}$	Bus Free Time between STOP and START Condition	SCL=1MHz	250	—	—	ns
t$_{LOW}$	SCL Low Time	SCL=1MHz	500	—	—	ns
t$_{HIGH}$	SCL High Time	SCL=1MHz	500	—	—	ns
t$_{su(DAT)}$	Setup Time SDA → SCL	SCL=1MHz	100	—	—	ns
t$_{su(STA)}$	START Condition Setup Time	SCL=1MHz	250	—	—	ns
t$_{su(STO)}$	STOP Condition Setup Time	SCL=1MHz	250	—	—	ns
t$_{h(DAT)}$	Hold Time SDA → SCL	SCL=1MHz	100	—	—	ns
t$_{h(STA)}$	START Condition Hold Time	SCL=1MHz	250	—	—	ns
t$_{r(SCL)}$	Rising Time of SCL Signal	SCL=1MHz	—	—	100	ns
t$_{f(SCL)}$	Falling Time of SCL Signal	SCL=1MHz	—	—	100	ns
t$_{r(SDA)}$	Rising Time of SDA Signal	SCL=1MHz	—	—	100	ns
t$_{f(SDA)}$	Falling Time of SDA Signal	SCL=1MHz	—	—	100	ns

Power-on Reset Characteristics

Ta=-40°C~85°C

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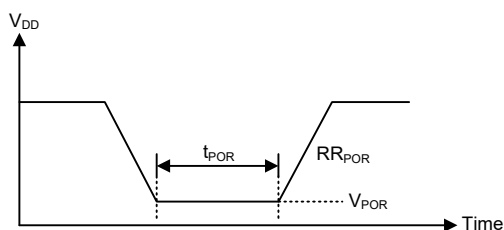
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, 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 an HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the decoding and execution functions. In this way, one T1~T4 clock cycle forms one instruction cycle. Although the fetching and execution of instructions takes place in consecutive instruction cycles, the pipelining structure of the microcontroller ensures that instructions are effectively executed in one instruction cycle. The exception to this are instructions where the contents of the Program Counter are changed, such as subroutine calls or jumps, in which case the instruction will take one more instruction cycle to execute.

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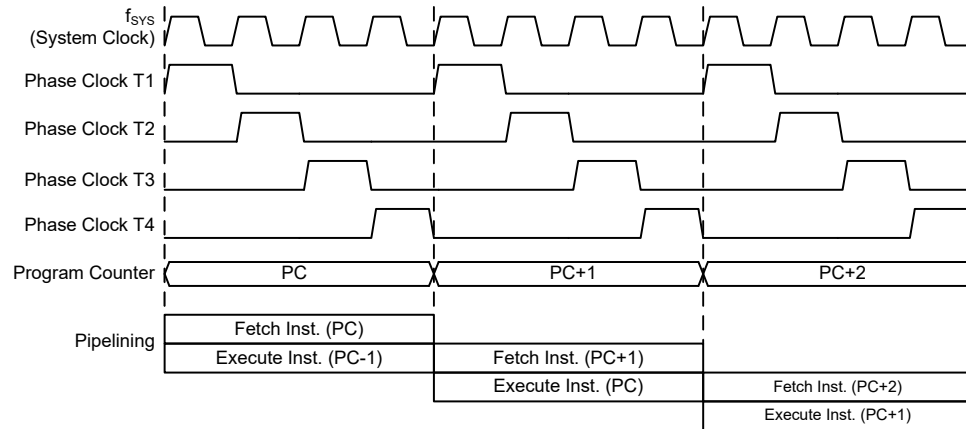
System Clocking and Pipelining

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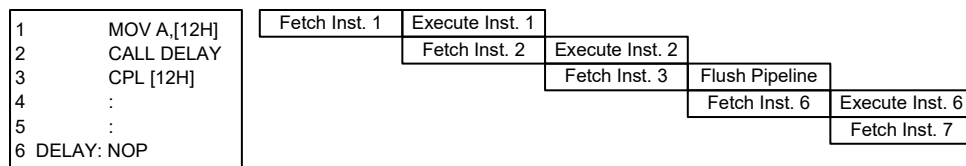
Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demand a jump to a non-consecutive Program Memory address. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

Program Counter	
High Byte	**Low Byte (PCL)**
PC10~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.
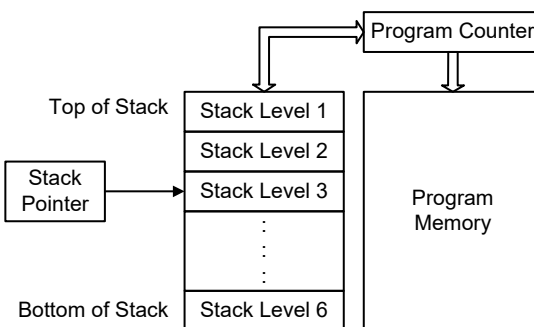
Arithmetic and Logic Unit – ALU

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
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement:
  INCA, INC, DECA, DEC
- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

Flash Program Memory

The Program Memory is the location where the user code or program is stored. For the device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, the Flash device 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×15 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.
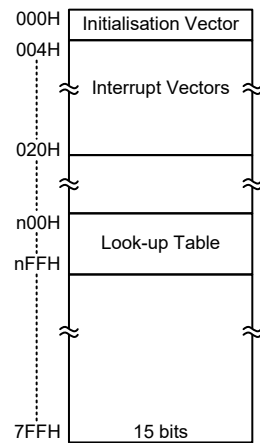
Program Memory Structure

Special Vectors

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

Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the "TABRD [m]" or "TABRDL [m]" instruction. 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.
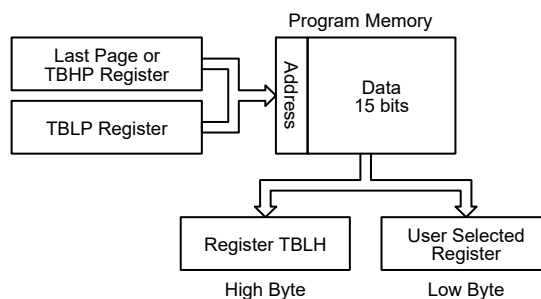

Table Program Example

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 "0700H" which refers to the start address of the last page within the 2K Program Memory of the device. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by the TBHP and TBLP registers if the "TABRD [m]" 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-only register and cannot be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the TBLH and subsequently cause errors if used again by the main routine. As a rule it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

Table Read Program Example

```
tempreg1 db ?          ; temporary register #1
tempreg2 db ?          ; temporary register #2
:
:
mov a, 06h             ; initialise low table pointer – note that
                       ; this 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
:
:
tabrd tempreg1         ; transfers value in table referenced by table
                       ; pointer data at program memory address "0706H"
                       ; 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 "0705H"
                       ; transferred to tempreg2 and TBLH in this example
                       ; the data "1AH" is transferred to tempreg1 and
                       ; data "0FH" to register tempreg2
```

```
        :
        :
org 0700h                ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
        :
        :
```

In Circuit Programming – ICP

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

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

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

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

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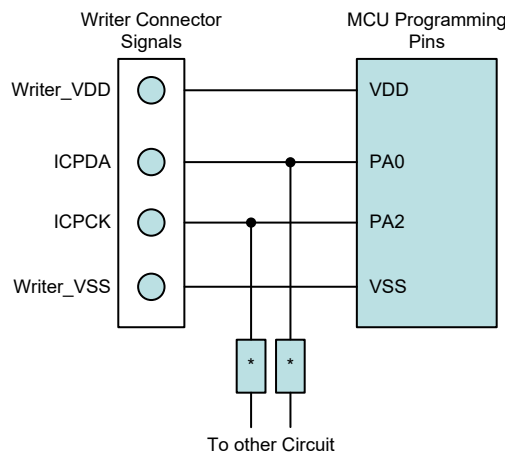
Note: * may be resistor or capacitor. The resistance of * must be greater than 1kΩ or the capacitance of * must be less than 1nF.

On-Chip Debug Support – OCDS

There is an EV chip named BC68V3132 which is used to emulate the BC68F3132 device. The EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip device for debugging, other pin functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

Data Memory

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

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

Structure

The overall Data Memory is subdivided into two banks. The Special Purpose Data Memory registers are accessible in all banks, with the exception of the EEC register at address 40H, which is only accessible in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory for the device is the address 00H.

Special Purpose Data Memory	General Purpose Data Memory	
Available Banks	Capacity	Bank: Address
Bank 0: 00H~7FH Bank 1: 40H (EEC only)	128×8	Bank 0: 80H~FFH

Data Memory Summary

Data Memory Structure
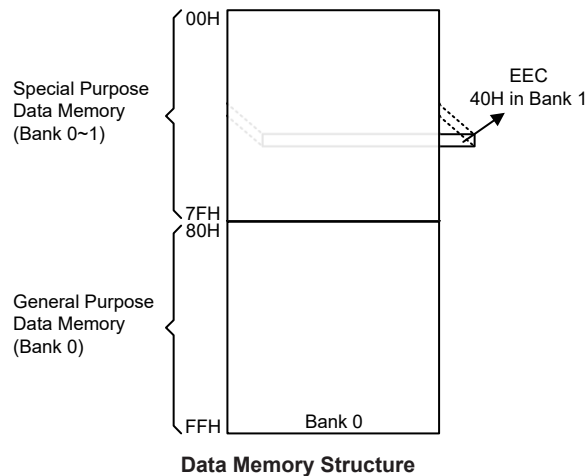
General Purpose Data Memory

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".

	Bank 0	Bank 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1	
04H	BP	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH		
0CH	INTEG	
0DH	WDTC	
0EH		
0FH	RSTFC	
10H		
11H	SCC	
12H	HIRCC	
13H		
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	PB	
19H	PBC	
1AH	PBPU	
1BH	LVDC	
1CH	LVRC	
1DH	PSCR	
1EH	EEA	
1FH	EED	
20H		
21H	TB0C	
22H	TB1C	
23H	INTC0	
24H	INTC1	
25H	INTC2	
26H	MFI0	
27H	MFI1	
28H	CTM0C0	
29H	CTM0C1	
2AH	CTM0DL	
2BH	CTM0DH	
2CH	CTM0AL	
2DH	CTM0AH	
2EH	CTM1C0	
2FH	CTM1C1	
30H	CTM1DL	
31H	CTM1DH	
32H	CTM1AL	
33H	CTM1AH	
34H	░░░░	
35H		
36H		
37H		
38H		
39H		
3AH		
3BH		
3CH		
3DH		
3EH		
3FH		

	Bank 0	Bank 1
40H		EEC
41H	HPEC	
42H	NLFSR3	
43H	NLFSR2	
44H	NLFSR1	
45H	NLFSR0	
46H	NLF3	
47H	NLF2	
48H	NLF1	
49H	NLF0	
4AH	HPKEY7	
4BH	HPKEY6	
4CH	HPKEY5	
4DH	HPKEY4	
4EH	HPKEY3	
4FH	HPKEY2	
50H	HPKEY1	
51H	HPKEY0	
52H	LOOPCNT1	
53H	LOOPCNT0	
54H	PAS0	
55H	PAS1	
56H	░░░░	
57H	PBS1	
58H	LVPUC	
59H	SLEWC	
5AH	DRVCC	
⋮		
7FH		

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

Special Purpose Data Memory Structure

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, 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 and IAR1 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 or MP1. Acting as a pair, IAR0 and MP0 can together access data only from Bank 0 while the IAR1 register together with the MP1 register can access data from any Data Memory Bank. 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, MP1

Two Memory Pointers, known as MP0 and MP1 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 Bank 0, while MP1 and IAR1 are used to access data from all banks according to the BP register. Direct Addressing can be used in Bank 0, all other banks must be addressed indirectly using MP1 and IAR1.

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

```
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:
```

The important point to note here is that in the examples shown above, no reference is made to specific Data Memory addresses.

Bank Pointer – BP

For this device, the Data Memory is divided into two banks, Bank 0 and Bank 1. Selecting the required Data Memory area is achieved using the Bank Pointer.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the SLEEP or IDLE Mode, in which case, the Data Memory bank remains unaffected. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from banks other than Bank 0 must be implemented using Indirect Addressing.

• **BP Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	DMBP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1   Unimplemented, read as "0"

Bit 0   **DMBP0**: Data Memory Bank selection
　　　　　　0: Bank 0
　　　　　　1: Bank 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 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 and C 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.

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	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R	R	R/W	R/W	R/W	R/W
POR	—	—	0	0	x	x	x	x

"x": Unknown

Bit 7~6   Unimplemented, read as "0"

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

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

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

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

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

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

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 64×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and a data register in Bank 0 and a single control register in Bank 1.

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Bank 1, can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEA	—	—	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM Register List

• EEA Register

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

Bit 7~6   Unimplemented, read as "0"

Bit 5~0   **EEA5~EEA0**: Data EEPROM address bit 5 ~ bit 0

• EED Register

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

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

• **EEC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	WREN	WR	RDEN	RD
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4   Unimplemented, read as "0"

Bit 3   **WREN**: Data EEPROM Write Enable
              0: Disable
              1: Enable
              This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

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

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

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

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

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

3. Ensure that the write operation is totally complete before changing the contents of the EEPROM related registers.

Reading Data from the EEPROM

To read data from the EEPROM, the EEPROM address of the data to be read must first be placed in the EEA register. The read enable bit, RDEN, in the EEC register must then be set high to enable the read function. If the RD bit in the EEC register is now set high, a read cycle will be initiated. Setting the RD bit high will not initiate a read operation if the RDEN bit has not been set. When the read cycle terminates, the RD bit will be automatically cleared to zero, after which the data can be read from the EED register. The data will remain in the EED register until another read or write operation is executed. The application program can poll the RD bit to determine when the data is valid for reading.

Writing Data to the EEPROM

To write data to the EEPROM, the EEPROM address of the data to be written must first be placed in the EEA register and the data placed in the EED register. To initiate a write cycle the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle successfully. These two instructions must be executed in two consecutive instruction cycles. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set high again after the write cycle has started. Note that setting the WR bit high will not initiate a write cycle if the WREN bit has not been set. As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended.

Write Protection

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

EEPROM Interrupt

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

Programming Considerations

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

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

Programming Examples

Reading data from the EEPROM – Polling Method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, 040H              ; setup memory pointer MP1
MOV MP1, A               ; MP1 points to EEC register
MOV A, 01H               ; setup Bank Pointer
MOV BP, A
SET IAR1.1               ; set RDEN bit, enable read operations
SET IAR1.0               ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0               ; check for read cycle end
JMP BACK
CLR IAR1                 ; disable EEPROM read if no more read operations are required
CLR BP
MOV A, EED               ; move read data to register
MOV READ_DATA, A
```

Note: For each read operation, the address register should be re-specified followed by setting the RD bit high to activate a read cycle even if the target address is consecutive.

Writing Data to the EEPROM – Polling Method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA       ; user defined data
MOV EED, A
MOV A, 040H              ; setup memory pointer MP1
MOV MP1, A               ; MP1 points to EEC register
MOV A, 01H               ; setup Bank Pointer
MOV BP, A
CLR EMI
SET IAR1.3               ; set WREN bit, enable write operations
SET IAR1.2               ; start Write Cycle - set WR bit - executed immediately
                        ; after set WREN bit
SET EMI
BACK:
SZ  IAR1.2               ; check for write cycle end
JMP BACK
CLR BP
```

Oscillators

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

Oscillator Overview

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

Type	Name	Frequency
Internal High Speed RC	HIRC	4/8/12MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two methods of generating the system clock, a high speed oscillator and a low speed oscillator. The high speed oscillator is the internal 4/8/12MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.
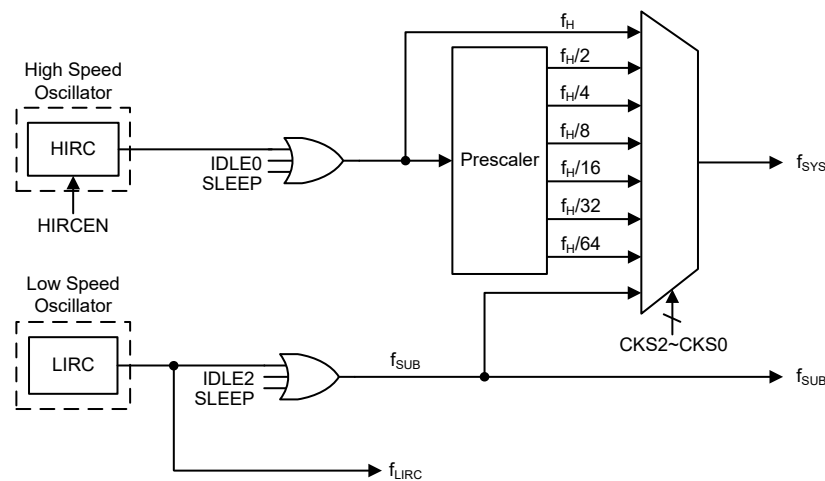
System Clock Configurations

Internal High Speed RC Oscillator – HIRC

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

Internal 32kHz Oscillator – LIRC

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

Operating Modes and System Clocks

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

System Clocks

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

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

Device Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed 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.

System Operation Modes

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

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

"x": Don't care

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

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

FAST Mode

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

SLOW Mode

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

SLEEP Mode

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

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN

System Operating Mode Control Register List

• **SCC Register**

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

Bit 7~5   **CKS2~CKS0**: System clock selection
    000: f_H
    001: $f_H/2$
    010: $f_H/4$
    011: $f_H/8$
    100: $f_H/16$
    101: $f_H/32$
    110: $f_H/64$
    111: f_{SUB}

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from f_H or f_{SUB}, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4~2   Unimplemented, read as "0"

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

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

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

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

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

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

• **HIRCC Register**

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

Bit 7~4   Unimplemented, read as "0"

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

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

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

Bit 1   **HIRCF**: HIRC oscillator stable flag
    0: 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 or the HIRC frequency selection is changed by the application program, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

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

Operating Mode Switching

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

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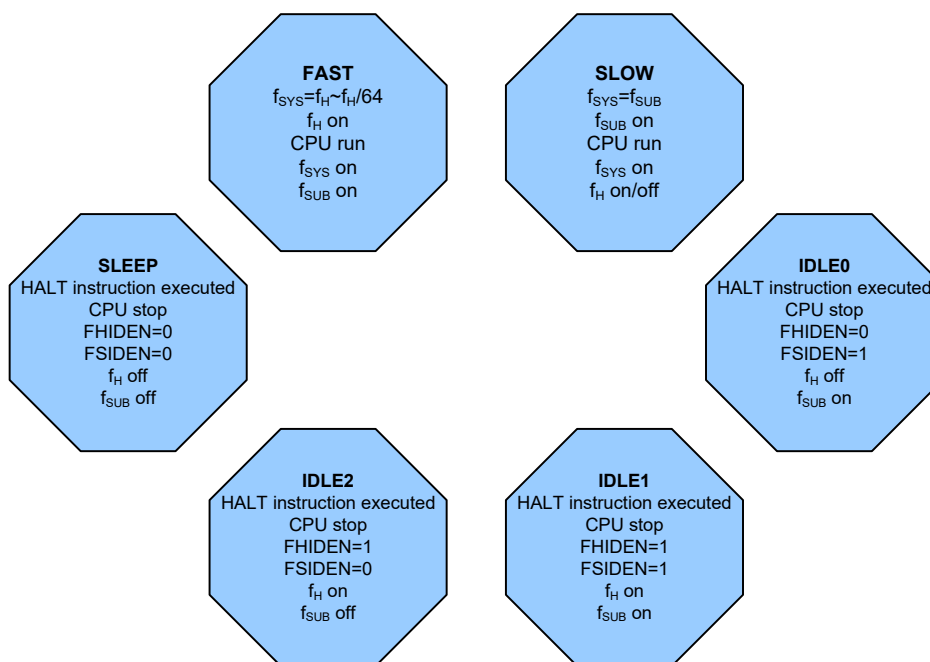
FAST Mode to SLOW Mode Switching

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

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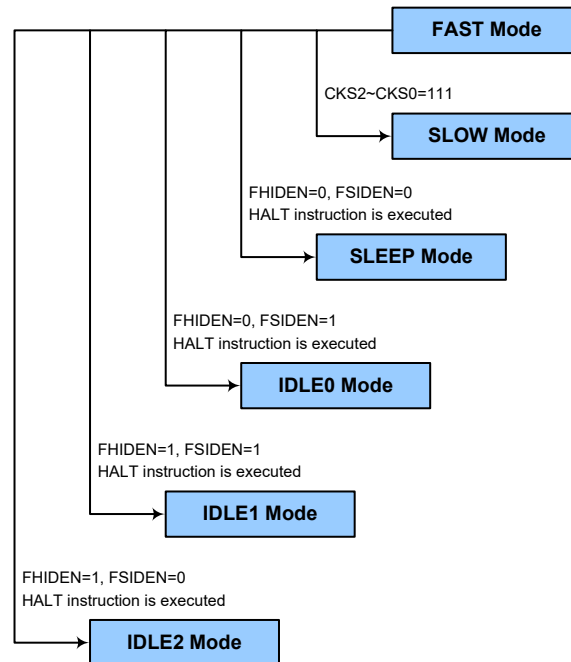
SLOW Mode to FAST Mode Switching

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

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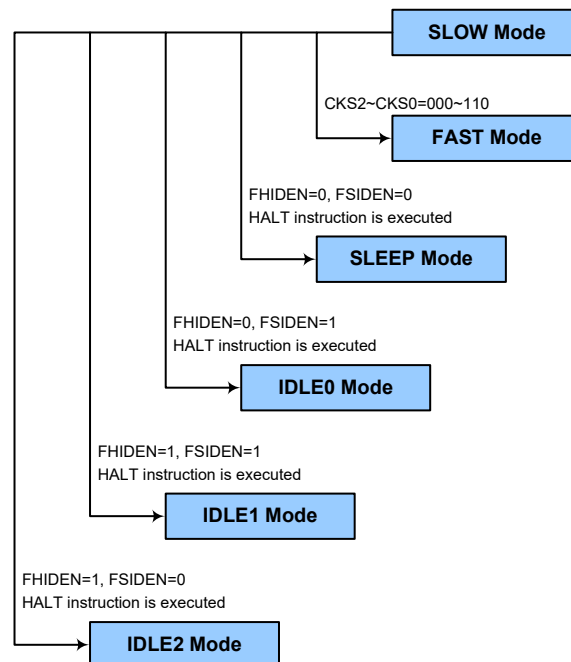
Entering the SLEEP Mode

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

• The system clock will be stopped and the application program will stop at the "HALT" instruction.

• The Data Memory contents and registers will maintain their present condition.

• The I/O ports will maintain their present conditions.

• In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.

• The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE0 Mode

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

• The f_H clock will be stopped and the application program will stop at the "HALT" instruction, but the f_{SUB} clock will be on.

• The Data Memory contents and registers will maintain their present condition.

• The I/O ports will maintain their present conditions.

• In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.

• The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE1 Mode

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

• The f_H and f_{SUB} clocks will be on but the application program will stop at the "HALT" instruction.

• The Data Memory contents and registers will maintain their present condition.

• The I/O ports will maintain their present conditions.

• In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.

• The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE2 Mode

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

• The f_H clock will be on but the f_{SUB} clock will be off and the application program will stop at the "HALT" instruction.

• The Data Memory contents and registers will maintain their present condition.

• The I/O ports will maintain their present conditions.

• In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.

• The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Standby Current Considerations

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

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

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

Wake-up

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

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

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

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

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{LIRC} which is sourced from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{15} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

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

• **WDTC Register**

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

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

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

Bit 2~0   **WS2~WS0**: WDT time-out period selection
        000: $2^8/f_{LIRC}$
        001: $2^9/f_{LIRC}$
        010: $2^{10}/f_{LIRC}$
        011: $2^{11}/f_{LIRC}$
        100: $2^{12}/f_{LIRC}$
        101: $2^{13}/f_{LIRC}$
        110: $2^{14}/f_{LIRC}$
        111: $2^{15}/f_{LIRC}$

        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	—	—	—	—	—	LVRF	LRF	WRF
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	x	0	0

"x": Unknown

Bit 7~3   Unimplemented, read as "0"

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

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

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

Watchdog Timer Operation

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

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

Watchdog Timer Function Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO and PDF bits in the status register will be set high and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 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^{15} 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 1 second for the 2^{15} division ratio, and a minimum timeout of 8ms for the 2^8 division ration.

Watchdog Timer
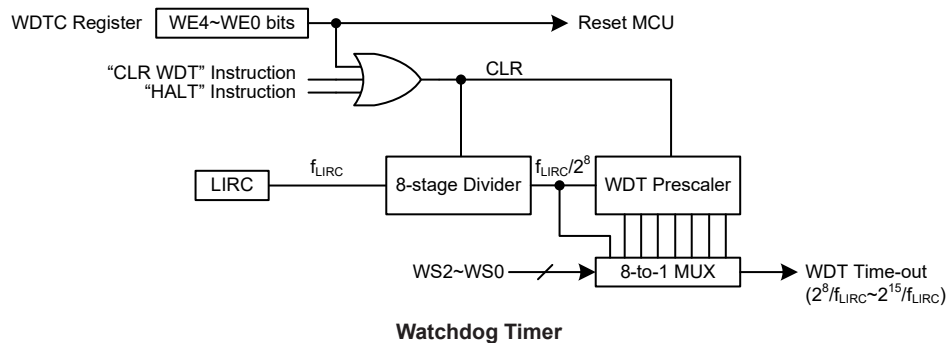
Reset and Initialisation

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

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

Reset Functions

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

Power-on Reset

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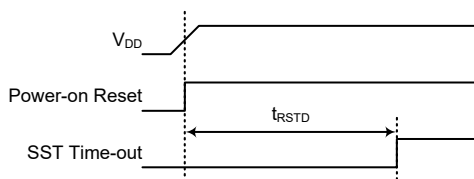
Power-on Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level. The LVR function can be enabled or disabled by the LVRC control register. If the LVRC control register is configured to enable the LVR function, the LVR function will be always enabled with a specific LVR voltage V_{LVR} in the FAST or SLOW mode. If the supply voltage of the device drops to within a range of $0.9V \sim V_{LVR}$ such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between $0.9V \sim V_{LVR}$ must exist for a time greater than that specified by t_{LVR} in the LVR/LVD Electrical Characteristics. If the 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 1.7V by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some different values, other than 01011010B and 10100101B, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01011010B. Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.
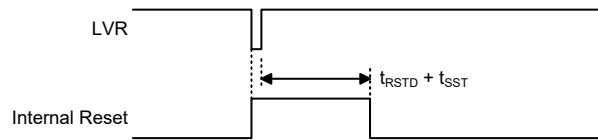
Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0   **LVS7~LVS0**: LVR voltage select
     01011010: 1.7V
     10100101: LVR disable
     Other values: Generates a MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by the defined LVR voltage value above, an MCU reset will 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 01011010B and 10100101B, 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	—	—	—	—	—	LVRF	LRF	WRF
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	x	0	0

                                                                       "x": Unknown

Bit 7~3   Unimplemented, read as "0"

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

Bit 1   **LRF**: LVR control register software reset flag
   0: Not occurred
     1: Occurred
     This bit is set to 1 by the LVRC control register contains any undefined LVR voltage register values. This in effect acts like a software-reset function. Note that 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.

Watchdog Time-out Reset during Normal Operation

When the Watchdog time-out Reset during normal operations in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to "1".
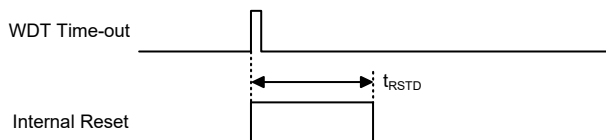
WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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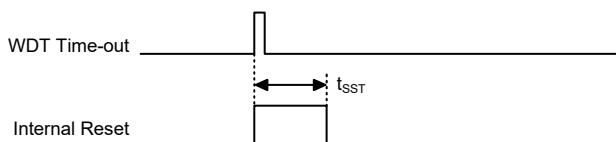
WDT Time-out Reset during SLEEP or IDLE Mode Timing Chart

Reset Initial Conditions

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

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

"u": Unchanged

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

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

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

Register	Power-on Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	xxxx xxxx	uuuu uuuu	uuuu uuuu
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu
IAR1	xxxx xxxx	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu
BP	---- ---0	---- ---0	---- ---u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu
TBHP	---- -xxx	---- -uuu	---- -uuu
STATUS	--00 xxxx	--1u uuuu	--11 uuuu
INTEG	---- --00	---- --00	---- --uu
WDTC	0101 0011	0101 0011	uuuu uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- 0001	---- 0001	---- uuuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PB	1--- --11	1--- --11	u--- --uu
PBC	1--- --11	1--- --11	u--- --uu
PBPU	0--- --00	0--- --00	u--- --uu
LVDC	--00 -000	--00 -000	--uu -uuu
LVRC	0101 1010	0101 1010	uuuu uuuu
PSCR	---- --00	---- --00	---- --uu
EEA	--00 0000	--00 0000	--uu uuuu
EED	0000 0000	0000 0000	uuuu uuuu
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	---0 ---0	---0 ---0	---u ---u
MFI0	--00 --00	--00 --00	--uu --uu
MFI1	--00 --00	--00 --00	--uu --uu
CTM0C0	0000 0000	0000 0000	uuuu uuuu
CTM0C1	0000 0000	0000 0000	uuuu uuuu
CTM0DL	0000 0000	0000 0000	uuuu uuuu
CTM0DH	---- --00	---- --00	---- --uu
CTM0AL	0000 0000	0000 0000	uuuu uuuu
CTM0AH	---- --00	---- --00	---- --uu
CTM1C0	0000 0000	0000 0000	uuuu uuuu
CTM1C1	0000 0000	0000 0000	uuuu uuuu

Register	Power-on Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
CTM1DL	0000 0000	0000 0000	uuuu uuuu
CTM1DH	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
CTM1AL	0000 0000	0000 0000	uuuu uuuu
CTM1AH	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
HPEC	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
NLFSR3	0000 0000	0000 0000	uuuu uuuu
NLFSR2	0000 0000	0000 0000	uuuu uuuu
NLFSR1	0000 0000	0000 0000	uuuu uuuu
NLFSR0	0000 0000	0000 0000	uuuu uuuu
NLF3	0000 0000	0000 0000	uuuu uuuu
NLF2	0000 0000	0000 0000	uuuu uuuu
NLF1	0000 0000	0000 0000	uuuu uuuu
NLF0	0000 0000	0000 0000	uuuu uuuu
HPKEY7	0000 0000	0000 0000	uuuu uuuu
HPKEY6	0000 0000	0000 0000	uuuu uuuu
HPKEY5	0000 0000	0000 0000	uuuu uuuu
HPKEY4	0000 0000	0000 0000	uuuu uuuu
HPKEY3	0000 0000	0000 0000	uuuu uuuu
HPKEY2	0000 0000	0000 0000	uuuu uuuu
HPKEY1	0000 0000	0000 0000	uuuu uuuu
HPKEY0	0000 0000	0000 0000	uuuu uuuu
LOOPCNT1	0000 0000	0000 0000	uuuu uuuu
LOOPCNT0	0000 0000	0000 0000	uuuu uuuu
PAS0	- - - - 0 0 - -	- - - - 0 0 - -	- - - - u u - -
PAS1	- - - - 0 0 - -	- - - - 0 0 - -	- - - - u u - -
PBS1	0 0 - - - - - -	0 0 - - - - - -	u u - - - - - -
LVPUC	- - - - - - - 0	- - - - - - - 0	- - - - - - - u
SLEWC	0000 0000	0000 0000	uuuu uuuu
DRVCC	- - - - 0000	- - - - 0000	- - - - uuuu
EEC	- - - - 0000	- - - - 0000	- - - - 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	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	—	—	—	—	—	PB1	PB0
PBC	PBC7	—	—	—	—	—	PBC1	PBC0
PBPU	PBPU7	—	—	—	—	—	PBPU1	PBPU0
LVPUC	—	—	—	—	—	—	—	LVPU

"—": Unimplemented, read as "0"

I/O Logic Function Register List

Note: For the interconnection lines, PB0 and PB1, their input/output, pull-high and other functions are also controlled by the corresponding bits in these registers. They should be properly configured to ensure correct operations. Refer to the relevant register description sections for more details.

Pull-high Resistors

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

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

• **PxPU Register**

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

PxPUn: I/O Port x Pin pull-high function control

  0: Disable

  1: Enable

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

Special attention has be paid to the PB0 and PB1 lines, which are internally connected to the RF transmitter inputs, DIN/SDA and PCLK/SCL, respectively, the corresponding PxPUn bits should be kept unchanged after power on.

• **LVPUC Register**

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

Bit 7~1 Unimplemented, read as "0"

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

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

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

    This bit is used to select the pull-high resistor value for low voltage power supply applications. Note that the LVPU bit is only available when the corresponding pin pull-high function is enabled by setting the relevant pull-high control bit high. This bit will have no effect when the pull-high function is disabled.

Port A Wake-up

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

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

• **PAWU Register**

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

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

    0: Disable

    1: Enable

I/O Port Control Registers

Each I/O port has its own control register known as PAC~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" is the Port name which can be A or B. However, the actual available bits for each I/O Port may be different.

Special attention has be paid to the PB0 and PB1 lines, which are internally connected to the RF transmitter inputs, DIN/SDA and PCLK/SCL, respectively, they should be configured as outputs after power on by clearing the corresponding PxCn bits, in order to properly control the RF transmitter.

I/O Port Output Source and Sink Current Selection

The I/O ports, PA~PB, can be setup to have a choice of various source and sink current using the DRVCC register. These source and sink current selection bits are available when the corresponding pin is configured as a CMOS output. Otherwise, these selection bits have no effect. Users should refer to the Input/Output Characteristics section to select the desired source and sink current for different applications.

• **DRVCC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	DRVCC3	DRVCC2	DRVCC1	DRVCC0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4   Unimplemented, read as "0"

Bit 3   **DRVCC3**: PB7 source & sink current selection
> 0: Source & Sink current = Level 0 (Min.)
> 1: Source & Sink current = Level 1 (Max.)

Bit 2   **DRVCC2**: The bit should be kept unchanged after power on

Bit 1   **DRVCC1**: PA7~PA4 source & sink current selection
> 0: Source & Sink current = Level 0 (Min.)
> 1: Source & Sink current = Level 1 (Max.)

Bit 0   **DRVCC0**: PA3~PA0 source & sink current selection
> 0: Source & Sink current = Level 0 (Min.)
> 1: Source & Sink current = Level 1 (Max.)

Note: Refer to the Input/Output Characteristics section to obtain the exact value.

I/O Port Output Slew Rate Selection

The I/O ports, PA~PB, can be setup to have a choice of various slew rate using the SLEWC register. These slew rate selection bits are available when the corresponding pin is configured as a CMOS output. Otherwise, these selection bits have no effect. Users should refer to the Input/Output Characteristics section to select the desired slew rate for different applications.

- **SLEWC Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEWC7	SLEWC6	SLEWC5	SLEWC4	SLEWC3	SLEWC2	SLEWC1	SLEWC0
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   **SLEWC7~SLEWC6**: PB7 output slew rate selection
         00: Slew rate = Level 0
         01: Slew rate = Level 1
         10: Slew rate = Level 2
         11: Slew rate = Level 3

Bit 5~4   **SLEWC5~SLEWC4**: These bits should be kept unchanged after power on

Bit 3~2   **SLEWC3~SLEWC2**: PA7~PA4 output slew rate selection
         00: Slew rate = Level 0
         01: Slew rate = Level 1
         10: Slew rate = Level 2
         11: Slew rate = Level 3

Bit 1~0   **SLEWC1~SLEWC0**: PA3~PA0 output slew rate selection
         00: Slew rate = Level 0
         01: Slew rate = Level 1
         10: Slew rate = Level 2
         11: Slew rate = Level 3

Note: Refer to the Input/Output Characteristics section to obtain the exact value.

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for digital input pins, such as INT, CTCKn, etc. which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bits. To

select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be set as an input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	—	—	—	—	PAS03	PAS02	—	—
PAS1	—	—	—	—	PAS13	PAS12	—	—
PBS1	PBS17	PBS16	—	—	—	—	—	—

Pin-shared Function Selection Register List

• **PAS0 Register**

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

Bit 7~4   Unimplemented, read as "0"

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

Bit 1~0   Unimplemented, read as "0"

• **PAS1 Register**

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

Bit 7~4   Unimplemented, read as "0"

Bit 3~2   **PAS13~PAS12**: PA5 pin-shared function selection
00: PA5
01: CTP1B
10: PA5
11: PA5

Bit 1~0   Unimplemented, read as "0"

• **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS17	PBS16	—	—	—	—	—	—
R/W	R/W	R/W	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

Bit 7~6   **PBS17~PBS16**: PB7 pin-shared function selection
00: PB7
01: CTP1
10: PB7
11: PB7

Bit 5~0   Unimplemented, read as "0"

I/O Pin Structure

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
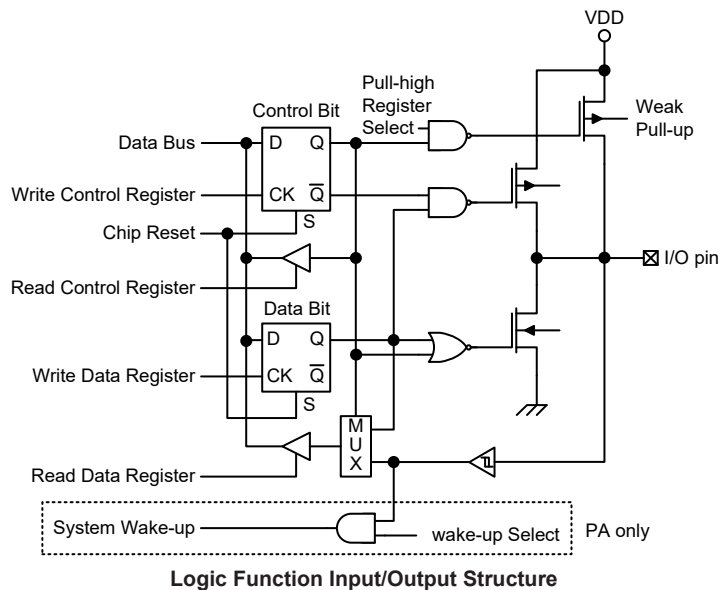
Programming Considerations

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

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

Timer Modules – TM

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

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

Introduction

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

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

TM Function Summary

TM Operation

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

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the CTnCK2~CTnCK0 bits in the CTMn control registers, where "n" stands for the specific TM serial number. The clock source can be a ratio of the system clock f_{SYS} or the internal high clock f_H, the f_{SUB} clock source or the external CTCKn pin. The CTCKn pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

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

TM External Pins

Each of the Compact Type TMs has one TM input pin, with the label CTCKn. The CTMn input pin, CTCKn, is essentially a clock source for the CTMn and is selected using the CTnCK2~CTnCK0 bits in the CTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The CTCKn input pin can be chosen to have either a rising or falling active edge.

The CTM0 has no external output pins while the CTM1 has two output pins with the label CTP1 and CTP1B. The CTP1B pin outputs the inverted signal of the CTP1. 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 CTP1 and CTP1B output pins are also the pins where the TM generates the PWM output waveform.

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

CTM	
Input	**Output**
CTCK0	—
CTCK1	CTP1, CTP1B

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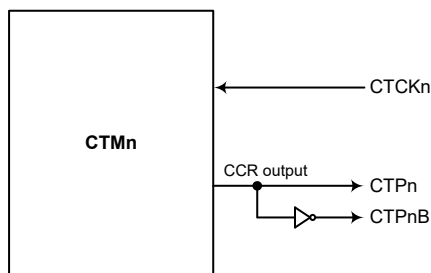
CTM Function Pin Block Diagram (n=0~1)

Programming Considerations

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

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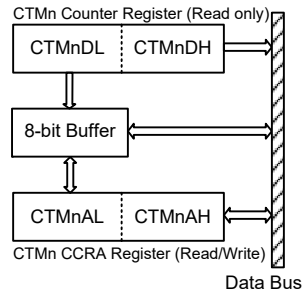
The following steps show the read and write procedures:

- Writing Data to CCRA
    - Step 1. Write data to Low Byte CTMnAL
        – Note that here data is only written to the 8-bit buffer.
    - Step 2. Write data to High Byte CTMnAH
        – Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.

- Reading Data from the Counter Registers and CCRA
    - Step 1. Read data from the High Byte CTMnDH and CTMnAH
        – Here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
    - Step 2. Read data from the Low Byte CTMnDL and CTMnAL
        – This step reads data from the 8-bit buffer.

Compact Type TM – CTM

The Compact Type TM contains three operating modes, which are Compare Match Output, Timer/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.
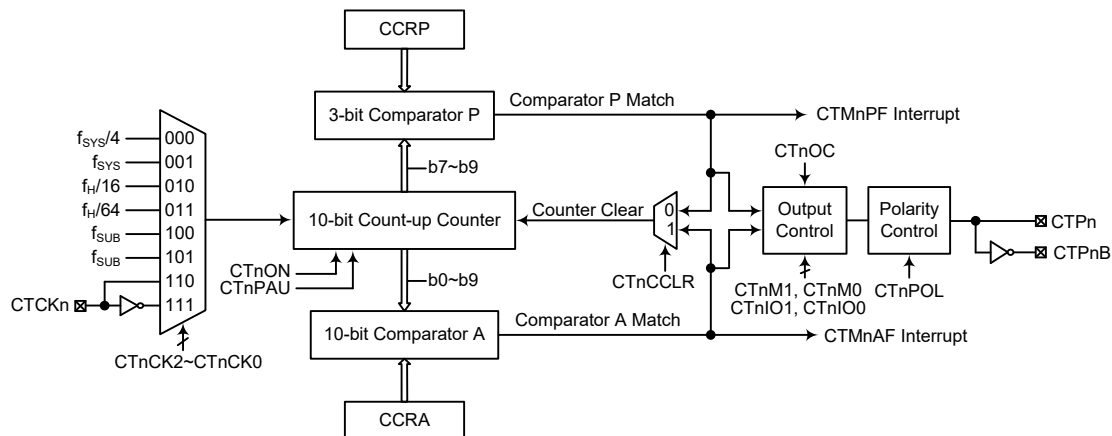
Note: 1. The CTMn external pins are pin-shared with other functions, therefore before using the CTMn function, ensure that the pin-shared function registers have been set properly to enable the CTMn pin function. The CTCKn pins, if used, must also be set as an input by setting the corresponding bits in the port control register.
2. The CTM0 has no external output pins.

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

Compact Type TM Operation

The size of Compact TM is 10-bit wide and its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP comparator is 3-bit wide whose value is compared with the highest 3 bits in the counter while the CCRA is the 10 bits and therefore compares all counter bits.

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

Compact Type TM Register Description

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

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

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

• **CTMnC0 Register**

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

Bit 7　　　**CTnPAU**: CTMn counter pause control

　　　　　0: Run
　　　　　1: Pause

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

Bit 6~4　　**CTnCK2~CTnCK0**: CTMn counter clock selection

　　　　　000: $f_{SYS}/4$
　　　　　001: f_{SYS}
　　　　　010: $f_H/16$
　　　　　011: $f_H/64$
　　　　　100: f_{SUB}
　　　　　101: f_{SUB}

110: CTCKn rising edge clock
111: CTCKn falling edge clock

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

Bit 3   **CTnON**: CTMn counter on/off control
0: Off
1: On

This bit controls the overall on/off function of the CTMn. Setting the bit high enables the counter to run while clearing the bit disables the CTMn. Clearing this bit to zero will stop the counter from counting and turn off the CTMn which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the CTMn is in the Compare Match Output Mode or the PWM Output Mode then the CTMn output pin will be reset to its initial condition, as specified by the CTnOC bit, when the CTnON bit changes from low to high.

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

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

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

- **CTMnC1 Register**

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

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

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

Bit 5~4   **CTnIO1~CTnIO0**: CTMn external pin function selection
Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

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

Timer/Counter Mode
 Unused

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

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

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

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

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

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

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

Bit 2 **CTnPOL**: CTMn CTPn output polarity control
 0: Non-invert
 1: Invert

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

Bit 1 **CTnDPX**: CTMn PWM duty/period control
 0: CCRP – period; CCRA – duty
 1: CCRP – duty; CCRA – period

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

Bit 0   **CTCCLR**: CTMn counter clear condition selection

0: Comparator P match

1: Comparator A match

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

• **CTMnDL Register**

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

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

CTMn 10-bit Counter bit 7 ~ bit 0

• **CTMnDH Register**

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

Bit 7~2   Unimplemented, read as "0"

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

CTMn 10-bit Counter bit 9 ~ bit 8

• **CTMnAL Register**

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

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

CTMn 10-bit CCRA bit 7 ~ bit 0

• **CTMnAH Register**

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

Bit 7~2   Unimplemented, read as "0"

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

CTMn 10-bit CCRA bit 9 ~ bit 8

Compact Type TM Operation Modes

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

Compare Match Output Mode

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

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

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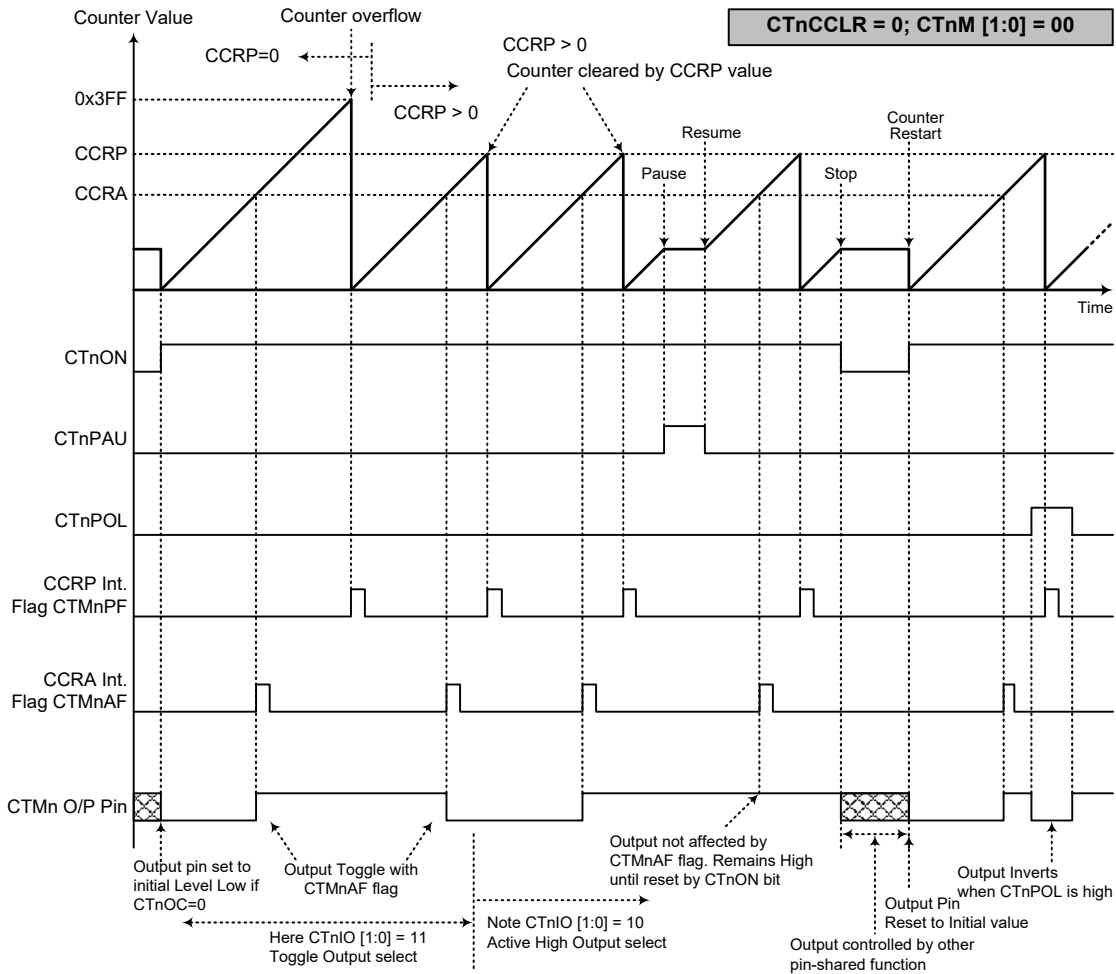
Compare Match Output Mode – CTnCCLR=0 (n=0~1)

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

Compare Match Output Mode – CTnCCLR=1 (n=0~1)
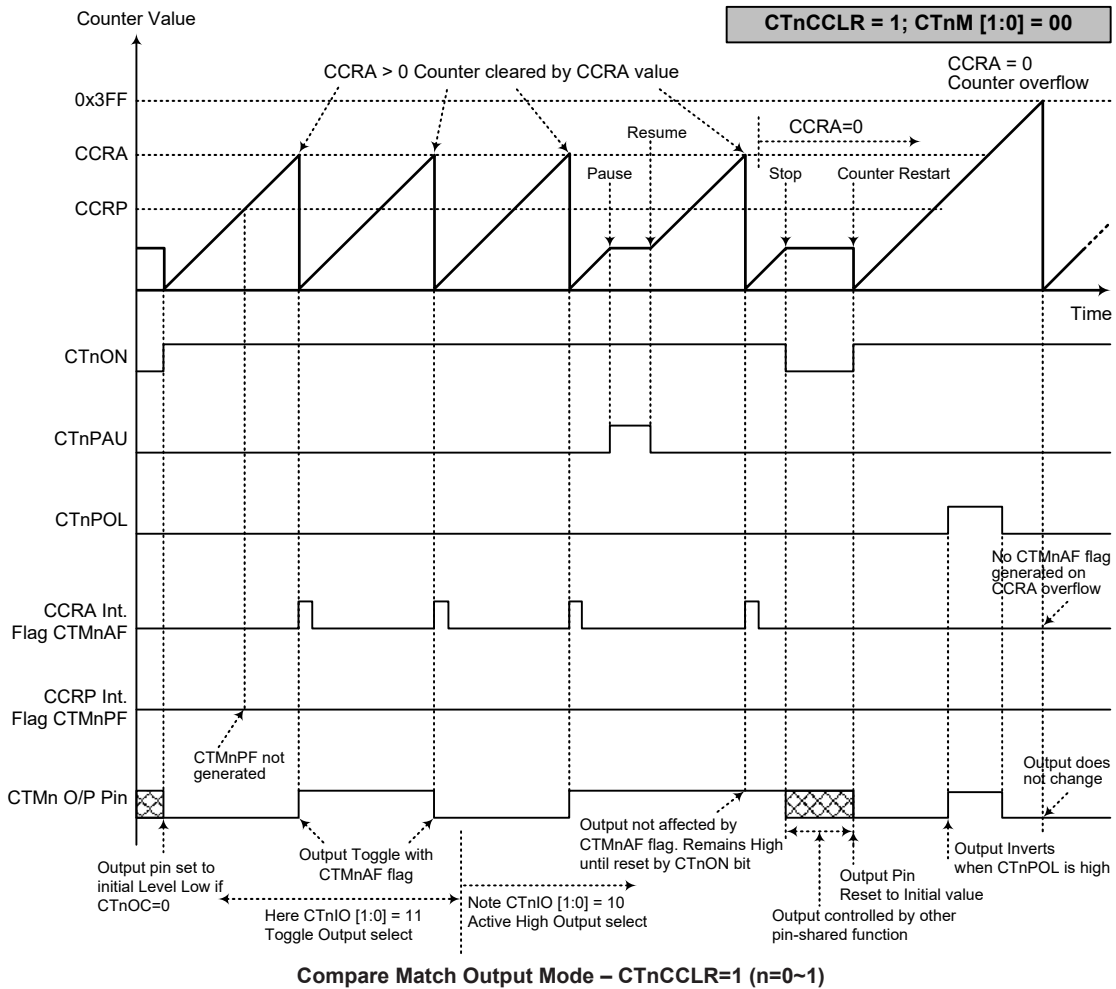
Note: 1. With CTnCCLR=1 a Comparator A match will clear the counter

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

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

4. A CTMnPF flag is not generated when CTnCCLR=1

Timer/Counter Mode

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

PWM Output Mode

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

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the CTnCCLR bit has no effect as the PWM period. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTnDPX bit in the CTMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The CTnOC bit in the CTMnC1 register is used to select the required polarity of the PWM waveform while the two CTnIO1 and CTnIO0 bits are used to enable the PWM output or to force the CTMn output pin to a fixed high or low level. The CTnPOL bit is used to reverse the polarity of the PWM output waveform.

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

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

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

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

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

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

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

The PWM output period is determined by the CCRA register value together with the CTMn clock while the PWM duty cycle is defined by the CCRP register value except when the CCRP value is equal to 0.

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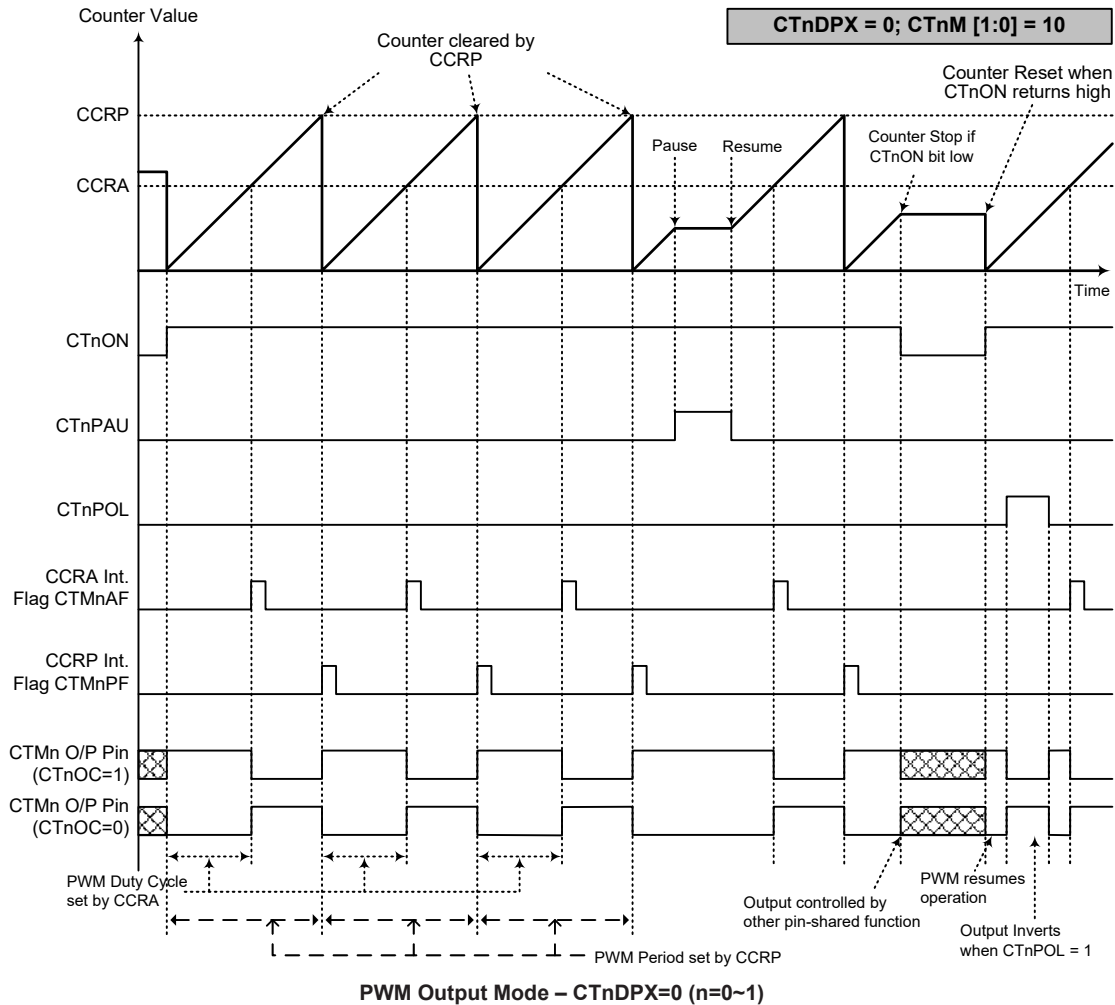
Note: 1. Here CTnDPX=0 – Counter cleared by CCRP

2. A counter clear sets the PWM Period

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

4. The CTnCCLR bit has no influence on PWM operation

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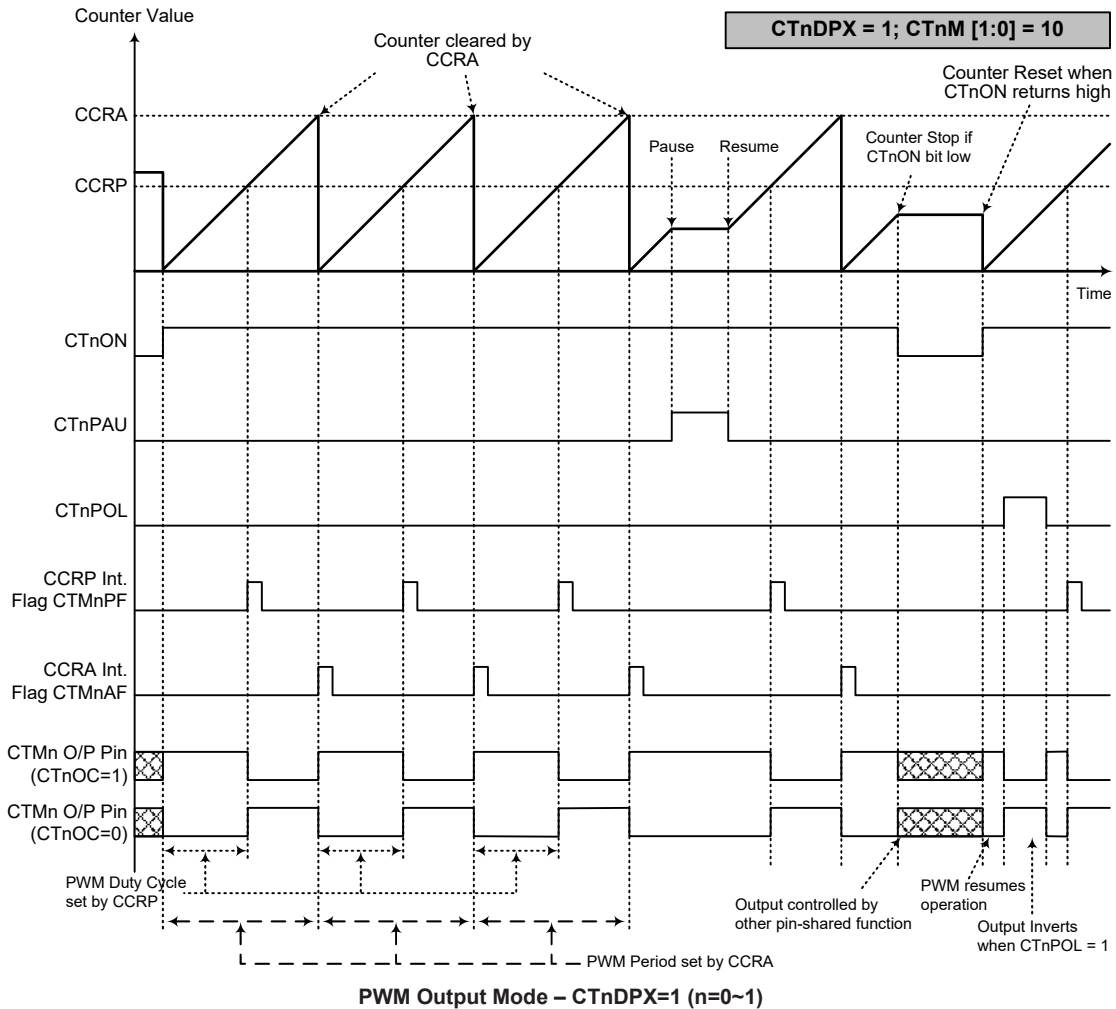
Note: 1. Here CTnDPX=1 – Counter cleared by CCRA

    2. A counter clear sets the PWM Period

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

    4. The CTnCCLR bit has no influence on PWM operation

Low Voltage Detector – LVD

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

LVD Register

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

• **LVDC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	—	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　　　**LVDEN**: Low voltage detector enable control
　　　　　　0: Dis/able
　　　　　　1: Enable

Bit 3　　　Unimplemented, read as "0"

Bit 2~0　　**VLVD2~VLVD0**: LVD voltage selection
　　　　　　000: 2.0V
　　　　　　001: 2.2V
　　　　　　010: 2.4V
　　　　　　011: 2.7V
　　　　　　100: 3.0V
　　　　　　101: 3.3V
　　　　　　110: 3.6V
　　　　　　111: $V_{LVDIN} \leq 1.25V$

When the VLVD2~VLVD0 bits are set to 111B, the LVD function will be implemented by comparing the LVDIN pin input voltage with the LVD reference voltage of 1.25V. When the VLVD2~VLVD0 bits are set to any other value except 111B, the LVD function will operate by comparing the V_{DD} voltage level with the LVD reference voltage with a specific voltage value which is generated by the internal LVD circuit.

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, or the LVDIN pin input voltage, with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.25V and 4.0V. When the power supply voltage, V_{DD}, or the LVDIN pin input voltage, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. When the device enters the SLEEP mode, the low voltage detector will be automatically disabled even if the LVDEN bit is set high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} or the LVDIN pin input voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.

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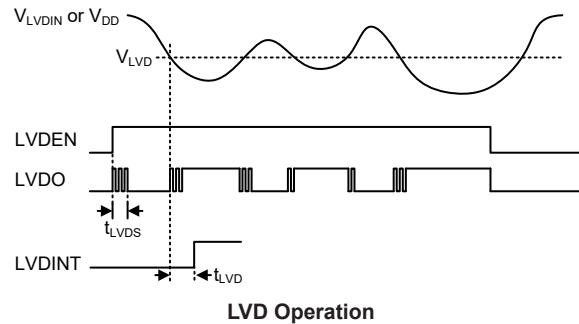
The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. In this case, the LVF interrupt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

Hopping Code Engine – HPE

The device includes a Hopping Code Engine, known as HPE, which is a Hopping code algorithm. It can operate in either the Encryption mode or Decryption mode, determined by the ENCDEC bit in the HPEC register. The Hopping Code Engine also has the capability of generating an interrupt signal.
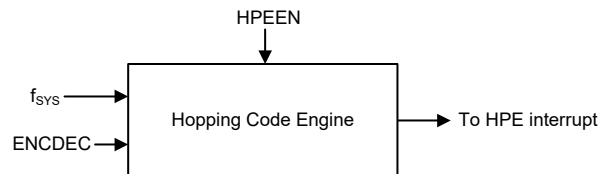
Hopping Code Engine Block Diagram

Hopping Code Engine Register Description

Overall operation of the Hopping Code Engine is controlled using a series of registers. A control register, HPEC, is used to control the enable/disable function and to select the operation mode. Two registers, LOOPCNT0~LOOPCNT1, exist to store the internal 16-bit loop down counter value. Eight registers, HPKEY0~HPKEY7, exist to store the 64-bit key shift value. Four registers, NLF0~NLF3, are used to store the 32-bit nonlinear-feedback value. The remaining four registers, NLFSR0~NLFSR3, are used to store the 32-bit nonlinear-feedback shift value.

Register Name	Bit							
	7	6	5	4	3	2	1	0
HPEC	—	—	—	—	—	—	ENCDEC	HPEEN
LOOPCNT0	D7	D6	D5	D4	D3	D2	D1	D0
LOOPCNT1	D15	D14	D13	D12	D11	D10	D9	D8
HPKEY0	D7	D6	D5	D4	D3	D2	D1	D0
HPKEY1	D15	D14	D13	D12	D11	D10	D9	D8
HPKEY2	D23	D22	D21	D20	D19	D18	D17	D16
HPKEY3	D31	D30	D29	D28	D27	D26	D25	D24
HPKEY4	D39	D38	D37	D36	D35	D34	D33	D32
HPKEY5	D47	D46	D45	D44	D43	D42	D41	D40
HPKEY6	D55	D54	D53	D52	D51	D50	D49	D48
HPKEY7	D63	D62	D61	D60	D59	D58	D57	D56
NLF0	D7	D6	D5	D4	D3	D2	D1	D0
NLF1	D15	D14	D13	D12	D11	D10	D9	D8
NLF2	D23	D22	D21	D20	D19	D18	D17	D16
NLF3	D31	D30	D29	D28	D27	D26	D25	D24
NLFSR0	D7	D6	D5	D4	D3	D2	D1	D0
NLFSR1	D15	D14	D13	D12	D11	D10	D9	D8
NLFSR2	D23	D22	D21	D20	D19	D18	D17	D16
NLFSR3	D31	D30	D29	D28	D27	D26	D25	D24

Hopping Code Engine Register List

- **HPEC Register**

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

Bit 7~2   Unimplemented, read as "0"

Bit 1   **ENCDEC**: Hopping code mode selection
   0: Encryption (Encoder)
   1: Decryption (Decoder)

Bit 0   **HPEEN**: Hopping code engine enable control
   0: Disable
   1: Enable – Execute algorithm
   When the HPEEN bit is set high by the software, if the LOOPCNT counter value is not equal to 0, the Hopping code algorithm will be executed. After the operation has completed, the HPEEN bit will be cleared to zero by hardware and an interrupt signal will also be generated.

- **LOOPCNT0 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**: Hopping code engine 16-bit loop down counter low byte

Note: 1. When the contents of the LOOPNT0~LOOPNT1 registers are equal to 0000h, the hopping code engine will be no operation.

   2. When the HPEEN bit is set high, the write function to the LOOPNT0~LOOPNT1 registers will be disabled.

- **LOOPCNT1 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**: Hopping code engine 16-bit loop down counter high byte

- **HPKEY0 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**: 64-bit Hopping code engine key shift register 0

Note: When the HPEEN bit is set high, the write function to the HPEKEY0~HPEKEY7 registers will be disabled.

- **HPKEY1 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**: 64-bit Hopping code engine key shift register 1

- **HPKEY2 Register**

Bit	7	6	5	4	3	2	1	0
Name	D23	D22	D21	D20	D19	D18	D17	D16
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   **D23~D16**: 64-bit Hopping code engine key shift register 2

- **HPKEY3 Register**

Bit	7	6	5	4	3	2	1	0
Name	D31	D30	D29	D28	D27	D26	D25	D24
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   **D31~D24**: 64-bit Hopping code engine key shift register 3

- **HPKEY4 Register**

Bit	7	6	5	4	3	2	1	0
Name	D39	D38	D37	D36	D35	D34	D33	D32
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   **D39~D32**: 64-bit Hopping code engine key shift register 4

• **HPKEY5 Register**

Bit	7	6	5	4	3	2	1	0
Name	D47	D46	D45	D44	D43	D42	D41	D40
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   **D47~D40**: 64-bit Hopping code engine key shift register 5

• **HPKEY6 Register**

Bit	7	6	5	4	3	2	1	0
Name	D55	D54	D53	D52	D51	D50	D49	D48
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   **D55~D48**: 64-bit Hopping code engine key shift register 6

• **HPKEY7 Register**

Bit	7	6	5	4	3	2	1	0
Name	D63	D62	D61	D60	D59	D58	D57	D56
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   **D63~D56**: 64-bit Hopping code engine key shift register 7

• **NLF0 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**: 32-bit Hopping code engine nonlinear-feedback register 0

Note: When the HPEEN bit is set high, the write function to the NLF0~NLF3 registers will be disabled.

• **NLF1 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**: 32-bit Hopping code engine nonlinear-feedback register 1

• **NLF2 Register**

Bit	7	6	5	4	3	2	1	0
Name	D23	D22	D21	D20	D19	D18	D17	D16
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   **D23~D16**: 32-bit Hopping code engine nonlinear-feedback register 2

• **NLF3 Register**

Bit	7	6	5	4	3	2	1	0
Name	D31	D30	D29	D28	D27	D26	D25	D24
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 **D31~D24**: 32-bit Hopping code engine nonlinear-feedback register 3

• **NLFSR0 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**: 32-bit Hopping code engine nonlinear-feedback shift register 0

Note: When the HPEEN bit is set high, the write function to the NLFSR0~NLFSR3 registers will be disabled.

• **NLFSR1 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**: 32-bit Hopping code engine nonlinear-feedback shift register 1

• **NLFSR2 Register**

Bit	7	6	5	4	3	2	1	0
Name	D23	D22	D21	D20	D19	D18	D17	D16
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 **D23~D16**: 32-bit Hopping code engine nonlinear-feedback shift register 2

• **NLFSR3 Register**

Bit	7	6	5	4	3	2	1	0
Name	D31	D30	D29	D28	D27	D26	D25	D24
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 **D31~D24**: 32-bit Hopping code engine nonlinear-feedback shift register 3
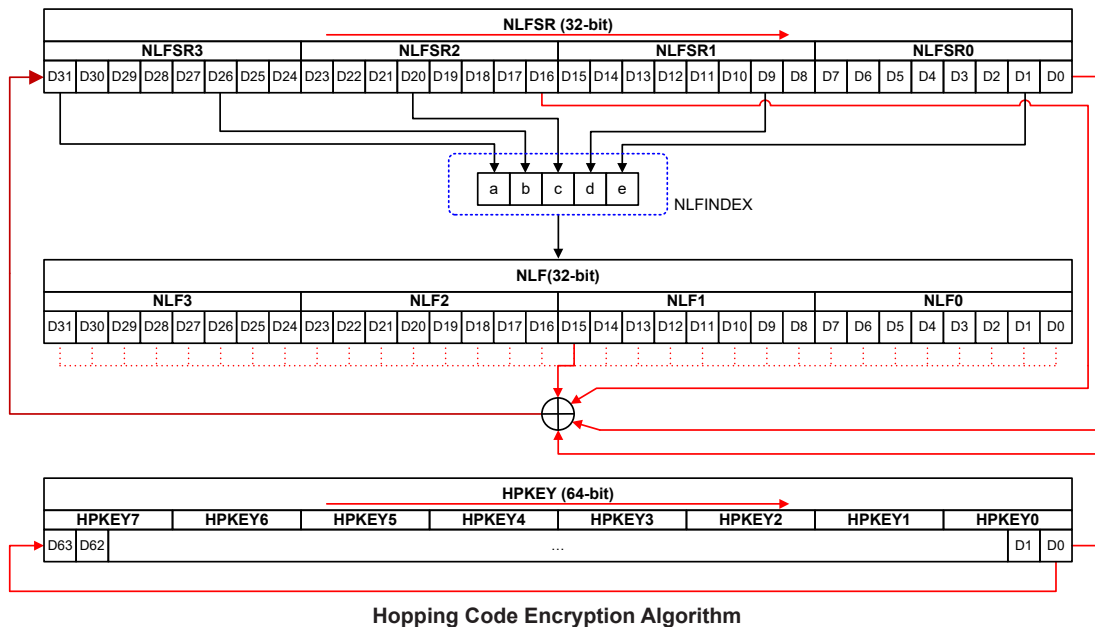
Hopping Code Engine Operation

For both Encryption and Decryption operations, three groups of register values, 64-bit HPKEY, 32-bit NLFSR and 32-bit NLF, together with a loop down counter value, 16-bit LOOPCNT, are used as the algorithm parameters.

Note that the device should not enter the IDLE or SLEEP mode until the Hopping code algorithm operations are totally completed. Otherwise the operations will fail.

Encryption Operation

To activate an Encryption operation, the HPEEN bit should first be set high to enable the Hopping code engine function and the ENCDEC bit should then be cleared to zero to select the Encryption mode. The following steps and diagram show the Encryption operation procedures:

- Step 1
  Check the LOOPCNT.
  If LOOPCNT≠0 (1~65535), then continue Step 2 ~ Step 5.
  If LOOPCNT=0, finish the Hooping code operation. However, no HPE interrupt signal will be generated. The HPEEN bit will be cleared to zero.

- Step 2
  Execute an XOR operation with the following four bits. These bits include NLFSR Bit 0, NLFSR Bit 16, HPKEY Bit 0 and one bit of NLF Bit 31 ~ Bit 0. This bit in the NLF is searched using the NLFINDEX value which is composed of NLFSR Bit 31, Bit 26, Bit 20, Bit 9 and Bit 1.

- Step 3
  Shift the NLFSR value right by one bit and move the operation result of Step 2 into the MSB.

- Step 4
  Shift the HPKEY value right by one bit, at which point HPKEY D63=D0.

- Step 5
  Run the LOOP once after the completion of Step 2 ~ Step 4. The loop down counter starts running and decreases the LOOPCNT value by 1.
  If LOOPCNT-1≠0, repeat Step 2 ~ Step 4.
  If LOOPCNT-1=0, finish the Hooping code operation. At this time, an HPE interrupt signal will be generated and the HPEEN bit will be cleared to zero by the hardware.

Hopping Code Encryption Algorithm

Decryption Operation

To activate a Decryption operation, the HPEEN bit should first be set high to enable the Hopping code engine function and the ENCDEC bit should then be set high to select the Decryption mode. The following steps and diagram show the Decryption operation procedures:

- Step 1

  Check the LOOPCNT.

  If LOOPCNT≠0 (1~65535), then continue Step 2 ~Step 6.

  If LOOPCNT=0, finish the Hooping code operation. However, no HPE interrupt signal will be generated. The HPEEN bit will be cleared to zero.

- Step 2

  Calculate (LOOPCNT-1)%64=n. For example, LOOPCNT=528, n=(528-1)%64=15.

  Note: The value of n does not vary with (LOOPCNT-1) in the operation, and is calculated only in the first LOOP.

- Step 3

  Execute an XOR operation with the following four bits. These bits include NLFSR Bit 31, NLFSR Bit 15, HPKEY Bit n and one bit of NLF Bit 31 ~ Bit 0. This bit in the NLF is searched using the NLFINDEX value which is composed of NLFSR Bit 30, Bit 25, Bit 19, Bit 8 and Bit 0.

- Step 4

  Shift the NLFSR value left by one bit and move the operation result of Step 3 into the LSB.

- Step 5

  Shift the HPKEY value left by one bit, at which point HPKEY D0=D63.

- Step 6

  Run the LOOP once after the completion of Step 3 ~ Step 5. The loop down counter starts running and decreases the LOOPCNT value by 1.

  If LOOPCNT-1≠0, repeat Step 3 ~ Step 5.

  If LOOPCNT-1=0, finish the Hooping code operation. At this time, an HPE interrupt signal will be generated and the HPEEN bit will be cleared to zero by the hardware.


Hopping Code Decryption Algorithm
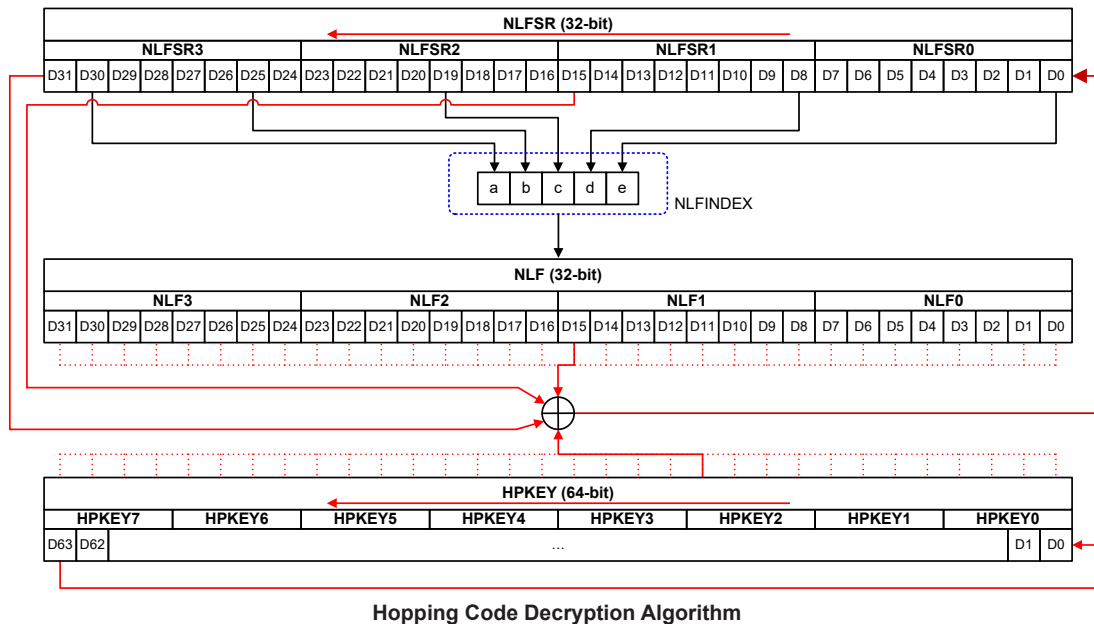
RF Transmitter

The RF transmitter is a cost effective Sub-1GHz OOK/FSK transmitter for wireless applications in the 315MHz, 433MHz, 470MHz, 868MHz and 915MHz frequency bands. It consists of a highly integrated fractional-N Synthesizer and a Class-E Power Amplifier (PA).

The RF frequency is generated by a fully integrated fractional-N Synthesizer which includes RF VCO, loop filter and Digital controlled XO (DCXO). A fractional-N synthesizer allows users to extend their applications to a wider frequency range with the same XO.

The transmit session is a VCO direct modulation architecture. Different from the conventional direct-up conversion transmitter, the FSK modulation signal is fed into the VCO directly to take advantage of fractional-N synthesizer. As a result, both layout area and current consumption are much smaller. The modulated signal, generated by VCO, is fed into a Class-E PA and the maximum output power can be up to +13dBm.

For OOK modulation applications, the RF transmitter provides an optimized PA ramping up and down feature to avoid the power spreading in the frequency domain.
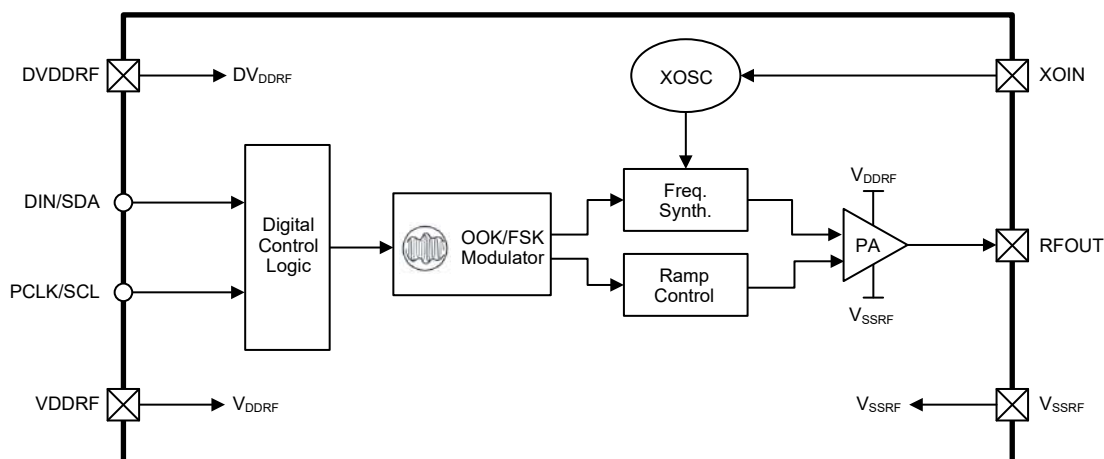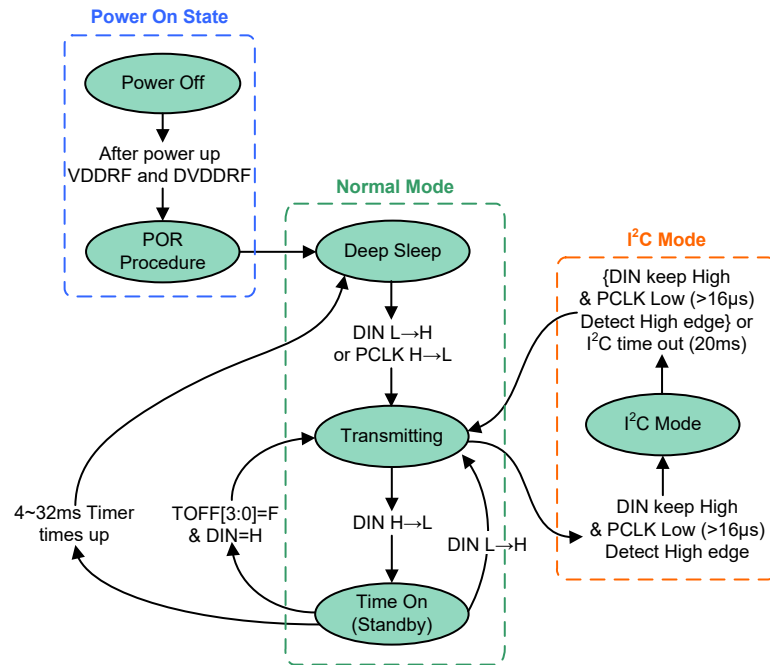

Note: The DIN/SDA and PCLK/SCL lines are internally connected to the MCU PB0 and PB1 lines respectively.

RF Transmitter Block Diagram

State Control

The RF transmitter has integrated state machines that control the state transition between modes.


Normal Mode

After a power-on reset operation, the RF transmitter will enter the Deep Sleep Mode to wait for a transmitting start condition. Data will be transmitted if the DIN line is pulled high or the pulse on the PCLK line changes from high to low. When data transmission is finished and the DIN line state changes from high to low, the RF transmitter will enter the Standby state and the Timer, whose timeout period is determined by the DLY_TOFF[3:0] bits in the CFG1 register, will turn on and start to count. The RF transmitter will return to the Deep Sleep Mode when the Timer overflows. However, it should be noted that when the DLY_TOFF[3:0] bit value is "1111", the RF transmitter will start to transmit again without entering the Deep Sleep Mode once the DIN line state changes from low to high.
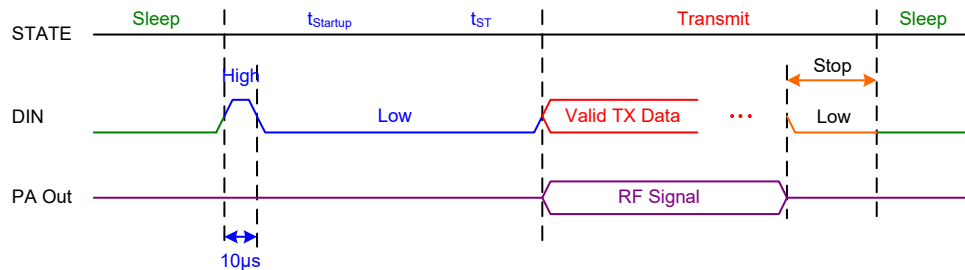
TX Enabled by DIN Line

I²C Mode

The RF transmitter can enter the I²C mode from the Normal mode. If the SCL line is pulled low for more than $16\mu s(t_{ENI2C})$, the RF transmitter will enter the I²C Mode for serial programming. When the RF transmitter receives a correct I²C STOP signal followed by the SCL line being pulled low for more than $16\mu s$, the RF transmitter will return to the Normal Mode.

Enter / Exit I²C Mode
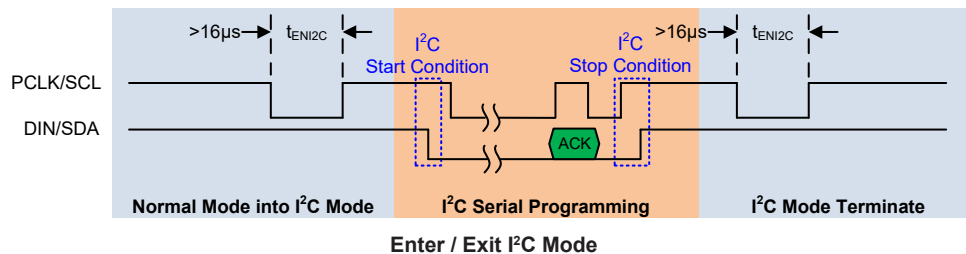
In the I²C Mode, the RF transmitter internal relevant registers can be configured using I²C serial programming. The transmitter only supports the I²C format for byte write, page write, byte read and page read format. The transmission procedure is shown below. It should be noted that the I²C is a non-standard I²C interface, which only supports a single device for connection.

Symbol Definition

- S: Start symbol
- RS: Repeat Start
- P: Stop symbol
- DADDR[6:0]: device address, 21h
- R/W: read write select, R(0): write, (1): read
- RADDR[7:0]: register address
- ACK: A(0):ACK, NA(1):NAK
- Bus Direction:

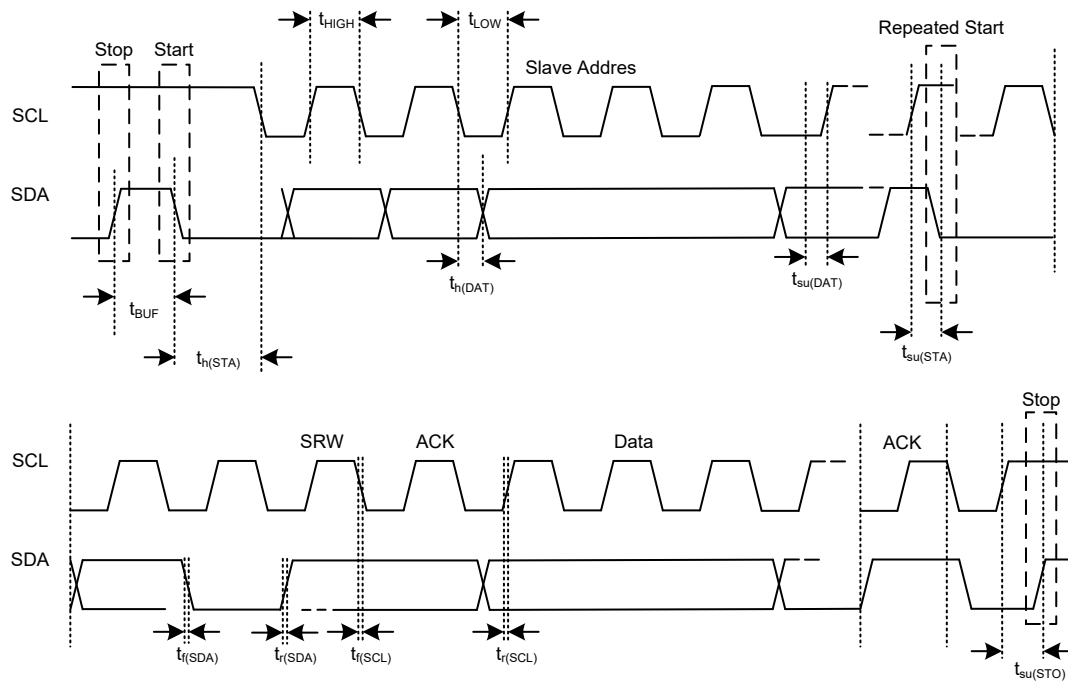 host to device:

  device to host:

Byte Write

Page Write

Byte Read

Page Read

I²C Communication Timing Diagram

RF Transmitter Registers

The RF transmitter can be setup and operated using a series of internal registers. The commands and data are written to and read from the RF transmitter using its internal I²C bus. This list provides a summary of all internal registers.

Address	Register Name	Bit							
		7	6	5	4	3	2	1	0
00h	CFG0	Setting0		XO_TRIM[5:0]					
01h	CFG1	DLY_TOFF[3:0]				Setting1			
02h	CFG2	FDEV[7:0]							
03h	CFG3	FSK_SEL	Setting2			TXPWR[3:0]			
04h	CFG4	D_N[5:0]						BAND_SEL[1:0]	
05h	CFG5	D_K[11:4]							
06h	CFG6	D_K[19:12]							
07h	CFG7	D7	Setting3						

The RF transmitter has a default state described as follows, determined by register initial values.

Modulation Mode: OOK

Operating Frequency: 433.92MHz

TX Output Power: 10dBm

XTAL Capacitor Load: 16.651pF

Power Off Delay Time: 32ms
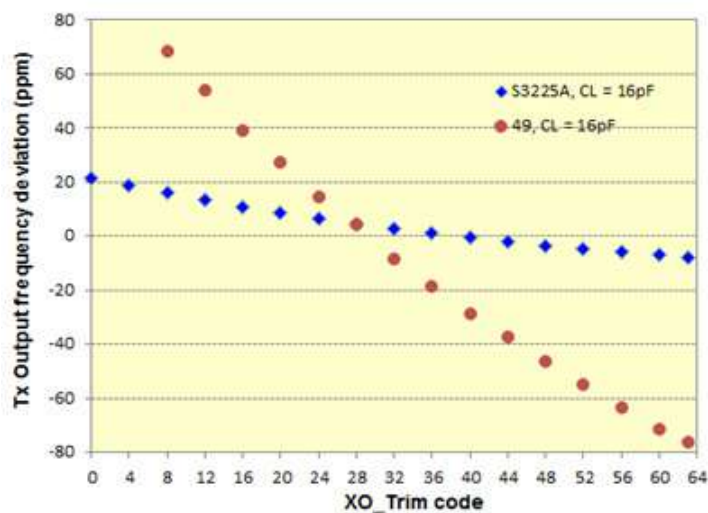
• **CFG0 – Configuration Control Register 0**

Address	Bit	7	6	5	4	3	2	1	0
00h	Name	Setting0		XO_TRIM[5:0]					
	R/W	R/W	R/W	R/W					
	Initial Value	1	0	1	0	0	0	0	0

Bit 7~6   **Setting0**: Must be "0b10"

Bit 5~0   **XO_TRIM[5:0]**: Trim value of the internal capacitor array for different crystal C_{LOAD}
Based on XO fabricated by YOKETAN corporation.
49US 16MHz XO with 16pF C_{LOAD}: The default setting is 1B. Within ±40ppm frequency error, 1 trim code shift -2.88ppm.
3225SMD 16MHz XO with 16pF C_{LOAD}: The default setting is 28. Within ±20ppm frequency error, 1 trim code shift -0.37ppm.

• **CFG1 – Configuration Control Register 1**

Address	Bit	7	6	5	4	3	2	1	0
01h	Name	DLY_TOFF[3:0]				Setting1			
	R/W	R/W				R/W			
	Initial Value	1	1	1	0	0	0	0	1

Bit 7~4   **DLY_TOFF[3:0]**: Transmitter Auto Power Off Delay Time
t=2ms×(DLY_TOFF[3:0]+2)
    0000: 4ms
    0001: 6ms
    0010: 8ms
    :
    1110: 32ms
    1111: Infinite – Never enter the Deep Sleep Mode

Bit 3~0   **Setting1**: Must be "0b0001"

• **CFG2 – Configuration Control Register 2**

Address	Bit	7	6	5	4	3	2	1	0
02h	Name				FDEV[7:0]				
	R/W				R/W				
	Initial Value	0	1	1	0	0	1	1	0

Bit 7~0   **FDEV[7:0]**: Frequency deviation for FSK

External Crystal=16MHz, FDEV=($f_{DEV} \times 2^{15}/f_{XTAL}$); f_{XTAL}=16MHz

Examples are as follows:

Default FDEV[7:0]=01100110→Decimal 102

External Crystal=16MHz

f_{DEV}(Frequency deviation)=$f_{DEV} \times$(16MHz/2^{15})

f_{DEV}(Frequency deviation)=102×(16MHz/32768)=49.8kHz

• **CFG3 – Configuration Control Register 3**

Address	Bit	7	6	5	4	3	2	1	0
03h	Name	FSK_SEL		Setting2			TXPWR[3:0]		
	R/W	R/W		R/W			R/W		
	Initial Value	0	1	0	0	1	0	0	0

Bit 7   **FSK_SEL**: FSK Mode Enable

  0: OOK
  1: FSK

Bit 6~4   **Setting 2**: Must be "0b100"

Bit 3~0   **TXPWR[3:0]**: RF Output Power

The RF transmitter has several output power values which are 0, 5, 10, and 13dBm.

TXPWR[3:0]	RF Output Power
0000	0dBm
0100	5dBm
1000	10dBm
1100	13dBm

TXPWR[3:0]	RF Output Power Fine Tune Level
xx00	0
xx01	1
xx10	2
xx11	3

Note that the adjust range: Level 3 > Level 2 > Level 1 > Level 0.

Note: Output power level could vary due to different matching components and placement on the PCB. The matching variation could significantly impact the output power level below +5dBm setting.

• **CFG4 – Configuration Control Register 4**

Address	Bit	7	6	5	4	3	2	1	0
04h	Name			D_N[5:0]				BAND_SEL[1:0]	
	R/W			R/W				R/W	
	Initial Value	0	1	0	1	1	0	0	1

Bit 7~2   **D_N[5:0]**: Integer of dividend for MMD

Bit 1~0   **BAND_SEL[1:0]**: Band Frequency Coarse Control

BAND_SEL	Frequency
00	315MHz
01	433MHz
10	868MHz
11	915MHz

Note that the BAND_SEL only select an approximate frequency range while the exact frequency value is determined by the D_N and D_K bit fields.

• **CFG5 – Configuration Control Register 5**

Address	Bit	7	6	5	4	3	2	1	0
05h	Name				D_K[11:4]				
	R/W				R/W				
	Initial Value	0	1	1	1	0	0	0	0

• **CFG6 – Configuration Control Register 6**

Address	Bit	7	6	5	4	3	2	1	0
06h	Name				D_K[19:12]				
	R/W				R/W				
	Initial Value	0	0	1	1	1	1	0	1

D_K[19:4]: 16-bit Fractional of dividend for MMD

D_N & D_K example.

XTAL = 16MHz and TX band = 433MHz

1. $D\_N \rightarrow (433M \times Divider)/16M = 54.125$

Take the integer part $\rightarrow D\_N = 54-32 = 22 \rightarrow 010110$

2. $D\_K \rightarrow (433M \times Divider)/16M = 54.125$

Take the fractional part $\rightarrow D\_K = 0.125 \times 2^{20} = 131072 \rightarrow 0010\text{-}0000\text{-}0000\text{-}0000$

3. The example frequency can be referred in the following table.

Band_SEL	Frequency	Divider	XTAL	D_N[5:0]	D_K[19:4]
315MHz	315MHz	2	16MHz	000111	0110-0000-0000-0000
433MHz	433MHz	2	16MHz	010110	0010-0000-0000-0000
433MHz	433.92MHz	2	16MHz	010110	0011-1101-0111-0000
868MHz	868MHz	1	16MHz	010110	0100-0000-0000-0000
915MHz	915MHz	1	16MHz	011001	0011-0000-0000-0000

• **CFG7 – Configuration Control Register 7**

Address	Bit	7	6	5	4	3	2	1	0
07h	Name	D7				Setting3			
	R/W	R				R/W			
	Initial Value	0	1	0	0	1	0	1	1

Bit 7   **D7**: Reserved bit

Bit 6~0   **Setting3**: Must be "0b1001011"

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, its corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to its needs. The device contains an external interrupt and several internal interrupt functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions such as the TMs, Time Bases, LVD and EEPROM, 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~INTC2 registers which setup the primary interrupts. The second is the MFI0~MFI1 register which setups the Multi-function interrupt. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INT Pin	INTE	INTF	—
Hopping code engine execution completed	HPEINTE	HPEINTF	—
Time Bases	TBnE	TBnF	n=0~1
CTM	CTMnPE	CTMnPF	n=0~1
	CTMnAE	CTMnAF	n=0~1
LVD	LVE	LVF	—
EEPROM write operation	DEE	DEF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	TB0F	HPEINTF	INTF	TB0E	HPEINTE	INTE	EMI
INTC1	LVF	MF1F	MF0F	TB1F	LVE	MF1E	MF0E	TB1E
INTC2	—	—	—	DEF	—	—	—	DEE
MFI0	—	—	CTM0AF	CTM0PF	—	—	CTM0AE	CTM0PE
MFI1	—	—	CTM1AF	CTM1PF	—	—	CTM1AE	CTM1PE

Interrupt Register List

- **INTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INTS1	INTS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2   Unimplemented, read as "0"

Bit 1~0   **INTS1~INTS0**: Interrupt edge control for INT 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	HPEINTF	INTF	TB0E	HPEINTE	INTE	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   **HPEINTF**: Hopping code engine execution completed interrupt request flag
　　　　　0: No request
　　　　　1: Interrupt request

Bit 4   **INTF**: INT interrupt request flag
　　　　　0: No request
　　　　　1: Interrupt request

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

Bit 2   **HPEINTE**: Hopping code engine execution completed interrupt control
　　　　　0: Disable
　　　　　1: Enable

Bit 1   **INTE**: INT 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	LVF	MF1F	MF0F	TB1F	LVE	MF1E	MF0E	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   **LVF**: LVD interrupt request flag
　　　　　0: No request
　　　　　1: Interrupt request

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

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

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

Bit 3   **LVE**: LVD interrupt control
    0: Disable
    1: Enable

Bit 2   **MF1E**: Multi-function interrupt 1 control
    0: Disable
    1: Enable

Bit 1   **MF0E**: Multi-function interrupt 0 control
    0: Disable
    1: Enable

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

• **INTC2 Register**

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

Bit 7~5   Unimplemented, read as "0"

Bit 4   **DEF**: Data EEPROM interrupt request flag
    0: No request
    1: Interrupt request

Bit 3~1   Unimplemented, read as "0"

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

• **MFI0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CTM0AF	CTM0PF	—	—	CTM0AE	CTM0PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6   Unimplemented, read as "0"

Bit 5   **CTM0AF**: CTM0 Comparator A match interrupt request flag
    0: No request
    1: Interrupt request

Bit 4   **CTM0PF**: CTM0 Comparator P match interrupt request flag
    0: No request
    1: Interrupt request

Bit 3~2   Unimplemented, read as "0"

Bit 1   **CTM0AE**: CTM0 Comparator A match interrupt control
    0: Disable
    1: Enable

Bit 0   **CTM0PE**: CTM0 Comparator P match interrupt control
   0: Disable
    1: Enable

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CTM1AF	CTM1PF	—	—	CTM1AE	CTM1PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **CTM1AF**: CTM1 Comparator A match interrupt request flag
 0: No request
 1: Interrupt request

Bit 4 **CTM1PF**: CTM1 Comparator P match interrupt request flag
 0: No request
 1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **CTM1AE**: CTM1 Comparator A match interrupt control
 0: Disable
 1: Enable

Bit 0 **CTM1PE**: CTM1 Comparator P match interrupt control
 0: Disable
 1: Enable

Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A 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 an "RETI", which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from

becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.

Interrupt Structure
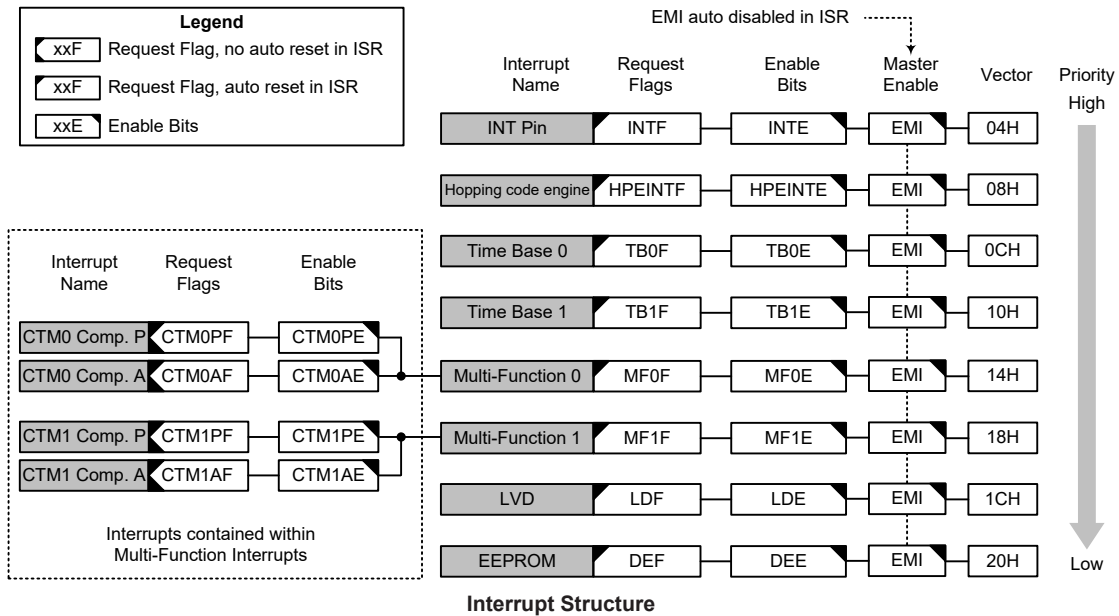
External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the external interrupt enable bit, INTE, 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 pin is pin-shared with I/O pin, it can only be configured as an external interrupt pin if its 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 set 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 flag, INTF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selection on the external interrupt pin 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.

Hopping Code Engine Interrupt

The Hopping Code Engine Interrupt is an individual interrupt source with its own interrupt vector. A Hopping Code Engine Interrupt request will take place when the Hopping Code Engine Interrupt request flag, HPEINTF, is set, which occurs when a hopping code engine execution is completed. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Hopping Code Engine Interrupt enable bit, HPEINTE, must first be set. When the interrupt is enabled, the stack is not full and a hopping code engine execution is completed, a subroutine call to the Hopping Code Engine Interrupt vector will take place. When the Hopping Code Engine Interrupt is serviced, the HPEINTF flag will be automatically cleared and the EMI bit will also be automatically cleared to disable other interrupts.

Time Base Interrupts

The function of the Time Base interrupts is to provide regular time signals in the form of an internal interrupt. It is controlled by the overflow signals from the timer function. When this happens its interrupt request flag, TBnF, will be set. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI and Time Base enable bit, TBnE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to its interrupt vector location will take place. When the interrupt is serviced, the interrupt request flag, TBnF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base interrupts is to provide an interrupt signal at fixed time periods. Its clock source, f_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBnC register to obtain longer interrupt periods whose value ranges. The clock source that generates f_{PSC}, which in turn controls the Time Base interrupt period is selected using the CLKSEL[1:0] bits in the PSCR register.
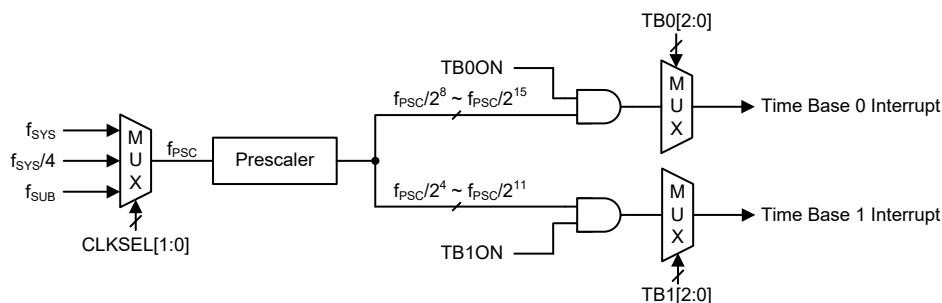
Time Base Interrupts

• **PSCR Register**

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

Bit 7~2   Unimplemented, read as "0"

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

• **TB0C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7   **TB0ON**: Time Base 0 control
         0: Disable
         1: Enable

Bit 6~3   Unimplemented, read as "0"

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

• **TB1C Register**

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

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

Bit 6~3   Unimplemented, read as "0"

Bit 2~0   **TB12~TB10**: Time Base 1 time-out period selection
         000: $2^4/f_{PSC}$
         001: $2^5/f_{PSC}$
         010: $2^6/f_{PSC}$
         011: $2^7/f_{PSC}$
         100: $2^8/f_{PSC}$
         101: $2^9/f_{PSC}$
         110: $2^{10}/f_{PSC}$
         111: $2^{11}/f_{PSC}$

Multi-function Interrupts

Within the device there are two Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts.

A Multi-function interrupt request will take place when the Multi-function interrupt request flag MFnF is set. The Multi-function interrupt flag will be set when any of its included functions generate an interrupt request flag. When the Multi-function interrupt is enabled and the stack is not full, and one of the interrupts contained within the Multi-function interrupt occurs, a subroutine call to the Multi-function interrupt vector will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function Interrupt request flag will be automatically reset when the interrupt is serviced, the request flag from the original source of the Multi-function interrupt will not be automatically reset and must be manually reset by the application program.

TM Interrupts

The Compact Type TMs each has two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. For each of the Compact Type TMs there are two interrupt request flags, CTMnPF and CTMnAF, and two enable control bits, CTMnPE and CTMnAE. 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 the relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant TM Interrupt vector locations will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

LVD Interrupt

The LVD Interrupt is an individual interrupt source with its own interrupt vector. An LVD interrupt request will take place when the LVD interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage or a low LVDIN pin input voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Low Voltage interrupt enable bit, LVE, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the LVD interrupt vector, will take place. When the LVD interrupt is serviced, the LVF flag will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

EEPROM Interrupt

The EEPROM Write Interrupt is an individual interrupt source with its own interrupt vector. An EEPROM Write Interrupt request will take place when the EEPROM Write Interrupt request flag, DEF, is set, which occurs when an EEPROM write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Write Interrupt enable bit, DEE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM write cycle ends, a subroutine call to the EEPROM Interrupt vector will take place. When the EEPROM Write Interrupt is serviced, the DEF flag will be automatically cleared and the EMI bit will also be automatically cleared to disable other interrupts.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pin or a low power supply voltage may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

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

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

Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

To return from an interrupt subroutine, either an RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into these devices during the programming process. During the development process, these options are selected using the HT-IDE software development tools. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Option	
1	HIRC Frequency Selection – f_{HIRC}: 4MHz, 8MHz or 12MHz

Note: When the HIRC has been configured at a frequency shown in this table, the HIRC1 and HIRC0 bits should also be setup to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Application Circuits
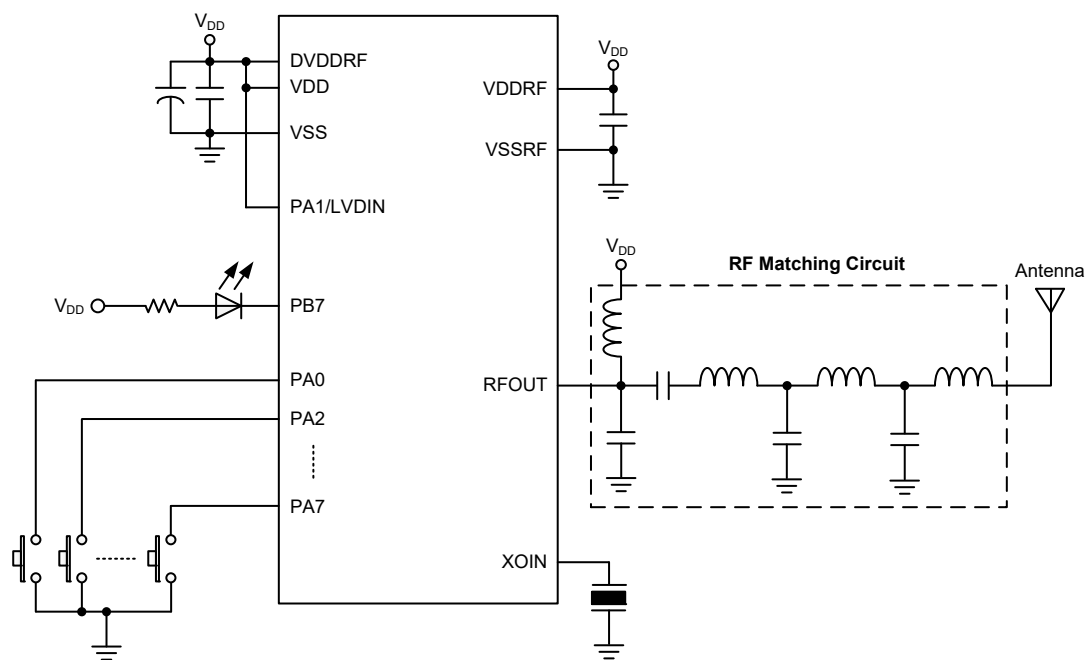
Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of three 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 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 set 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 following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

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
ADDM A,[m]	Add ACC to Data Memory	1[Note]	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1[Note]	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1[Note]	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1[Note]	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1[Note]	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1[Note]	Z
ORM A,[m]	Logical OR ACC to Data Memory	1[Note]	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1[Note]	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1[Note]	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1[Note]	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1[Note]	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1[Note]	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1[Note]	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1[Note]	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1[Note]	C

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1[Note]	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1[Note]	None
SET [m].i	Set bit of Data Memory	1[Note]	None
Branch Operation			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1[Note]	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1[Note]	None
SZ [m].i	Skip if bit i of Data Memory is zero	1[Note]	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1[Note]	None
SIZ [m]	Skip if increment Data Memory is zero	1[Note]	None
SDZ [m]	Skip if decrement Data Memory is zero	1[Note]	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1[Note]	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1[Note]	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read Operation			
TABRD [m]	Read table (specific page or current page) to TBLH and Data Memory	2[Note]	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2[Note]	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1[Note]	None
SET [m]	Set Data Memory	1[Note]	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1[Note]	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

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

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

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

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

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

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	$[m] \leftarrow [m] - 1$
Affected flag(s)	Z

DECA [m]   Decrement Data Memory with result in ACC

Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow [m] - 1$
Affected flag(s)	Z

HALT   Enter power down mode

Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.
Operation	$TO \leftarrow 0$ $PDF \leftarrow 1$
Affected flag(s)	TO, PDF

INC [m]   Increment Data Memory

Description	Data in the specified Data Memory is incremented by 1.
Operation	$[m] \leftarrow [m] + 1$
Affected flag(s)	Z

INCA [m]   Increment Data Memory with result in ACC

Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow [m] + 1$
Affected flag(s)	Z

JMP addr   Jump unconditionally

Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter $\leftarrow$ addr
Affected flag(s)	None

MOV A,[m]   Move Data Memory to ACC

Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	$ACC \leftarrow [m]$
Affected flag(s)	None

MOV A,x   Move immediate data to ACC

Description	The immediate data specified is loaded into the Accumulator.
Operation	$ACC \leftarrow x$
Affected flag(s)	None

MOV [m],A   Move ACC to Data Memory

Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	$[m] \leftarrow ACC$
Affected flag(s)	None

NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None

OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ x
Affected flag(s)	Z

ORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″OR″ [m]
Affected flag(s)	Z

RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None

RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None

RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None

RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← [m].7
Affected flag(s)	None

RLA [m]	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← [m].7
Affected flag(s)	None

RLC [m]	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← C C ← [m].7
Affected flag(s)	C

RLCA [m]	Rotate Data Memory left through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← C C ← [m].7
Affected flag(s)	C

RR [m]	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← [m].0
Affected flag(s)	None

RRA [m]	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [m].(i+1); (i=0~6) ACC.7 ← [m].0
Affected flag(s)	None

RRC [m]	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← C C ← [m].0
Affected flag(s)	C

RRCA [m]	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC.i \leftarrow [m].(i+1)$; $(i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C

SBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C

SBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C

SDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] - 1$ Skip if $[m]=0$
Affected flag(s)	None

SDZA [m]	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m] - 1$ Skip if $ACC=0$
Affected flag(s)	None

SET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	$[m] \leftarrow FFH$
Affected flag(s)	None

SET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	$[m].i \leftarrow 1$
Affected flag(s)	None

SIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] + 1 Skip if [m]=0
Affected flag(s)	None

SIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] + 1 Skip if ACC=0
Affected flag(s)	None

SNZ [m].i	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None

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

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

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

SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0 ↔ [m].7~[m].4
Affected flag(s)	None

SWAPA [m]   Swap nibbles of Data Memory with result in ACC

Description   The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation   ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s)   None

SZ [m]   Skip if Data Memory is 0

Description   The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation   Skip if [m]=0

Affected flag(s)   None

SZA [m]   Skip if Data Memory is 0 with data movement to ACC

Description   The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation   ACC ← [m]
Skip if [m]=0

Affected flag(s)   None

SZ [m].i   Skip if bit i of Data Memory is 0

Description   If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation   Skip if [m].i=0

Affected flag(s)   None

TABRD [m]   Read table (specific page or current page) to TBLH and Data Memory

Description   The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no 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

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

Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)

- The Operation Instruction of Packing Materials

- Carton information

16-pin NSOP-EP (150mil) Outline Dimensions

THERMAL VARIATIONS ONLY
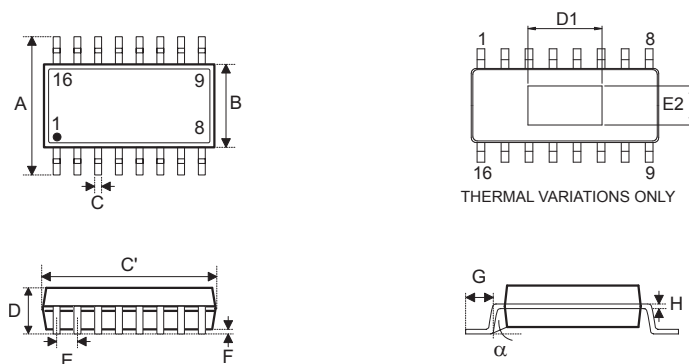
Symbol	Dimensions in inch		
	Min.	**Nom.**	**Max.**
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C'	—	0.390 BSC	—
D	—	—	0.069
D1	0.152	—	0.186
E	—	0.050 BSC	—
E2	0.066	—	0.101
F	0.000	—	0.006
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

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	3.86	—	4.72
E	—	1.27 BSC	—
E2	1.68	—	2.56
F	0.00	—	0.15
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

SAW Type 16-pin QFN (3mm×3mm for FP0.25mm) Outline Dimensions
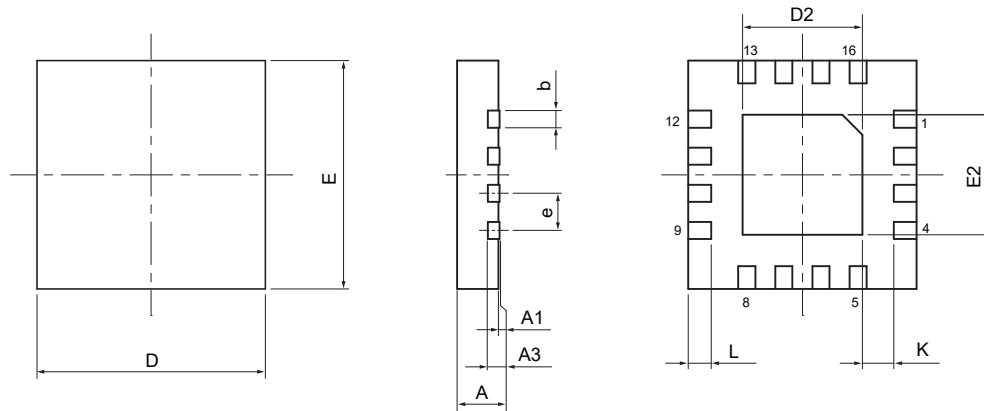
Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	—	0.008 REF	—
b	0.007	0.010	0.012
D	—	0.118 BSC	—
E	—	0.118 BSC	—
e	—	0.020 BSC	—
D2	0.063	—	0.069
E2	0.063	—	0.069
L	0.008	0.010	0.012
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	—	0.203 REF	—
b	0.18	0.25	0.30
D	—	3.00 BSC	—
E	—	3.00 BSC	—
e	—	0.50 BSC	—
D2	1.60	—	1.75
E2	1.60	—	1.75
L	0.20	0.25	0.30
K	0.20	—	—