BLE Beacon Body Fat Measurement Flash MCU

BH66F71652/BH66F71662

Revision: V1.00 Date: January 10, 2020

www.holtek.com

Table of Contents

Features

CPU Features

- Operating voltage
  - f_{SYS}=8MHz: 2.2V~3.6V
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 8MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 8K×16~16K×16
- Data Memory: 384×8~512×8
- True EEPROM Memory: 32×8~64×8
- Watchdog Timer function
- 17 bidirectional I/O lines
- Programmable I/O source current for LED applications
- 2 pin-shared external interrupts
- Up to two Timer Modules for time measurement, input capture, compare match output or PWM output or single pulse output function
- Serial Peripheral Interface – SPI
- Fully-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- Dual Time-Base functions for generation of fixed time interrupt signals
- 2 differential or 4 single-end channel 24-bit resolution Delta Sigma A/D converter
- Internal LDO with bypass function for PGA, ADC and external sensor power supply
- Body Fat Measurement Circuit
- Low voltage reset function
- Low voltage detect function
- Package type: 46-pin QFN

RF Transmitter Features

- Supports GFSK (BT=0.5) modulation with 1Mbps data rate, compliant with BLE standard
- Operating frequency: 2402/2426/2480MHz
- Programmable TX power
  - High power matching: -10/-5/-2/+5dBm (Max. +8dBm)
  - Low power matching: -10/-5/0/+2dBm (Max. +5dBm)

- Low current consumption
  - Low deep sleep current: 0.35μA
  - TX current consumption: 19mA @ 5dBm TX power
  - TX current consumption: 12mA @ -5dBm TX power
- Supports 32MHz crystal
- FCC/ETSI compliant

Applications

BLE Beacon transmission Body Fat Meter

General Description

The devices are specifically designed for Bluetooth Beacon transmission four-electrode AC Body Fat Scale applications. Measuring body fat uses a technique whereby an AC current flowing through the human body is measured and then used to calculate a body fat value. The specialised circuits to do this are a weight measurement circuit and a fat measurement circuit. The weight measurement circuit uses an external load cell to output a signal, which after amplification by an operational amplifier, and then conversion using an A/D converter, reads the corresponding value as the calculated weight. The fat measurement circuit uses an AC signal via an electrode slice to flow through human body. After amplification by an internal operational amplifier, and then conversion by an A/D converter, the measured value is one representing body impedance, which is used to calculate the corresponding body fat value.

The devices are Flash Memory I/O type 8-bit high performance RISC architecture microcontrollers which integrates a four-electrode LED Bluetooth Beacon transmission circuit and a multi-channel 24-bit Delta Sigma A/D converter, designed for applications that interface directly to analog signals and which require a low noise and high accuracy analog-to-digital converter.

Offering users the convenience of Flash Memory multi-programming features, the devices also include a wide range of functions and features. Other memory includes an area of Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 24-bit Delta Sigma A/D converter and other circuitry specifically designed for Body Fat Scale applications. Up to two extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is provided by including fully integrated SPI and UART interface functions, two popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

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

Bluetooth Beacon adopts BLE advertising feature to support new applications such as indoor navigation, healthcare, etc. The advertised packets are placed in the Channel 37, 38 and 39 to avoid the interference from WiFi channels at ISM band. The BH66F716x2 is aimed for the Beacon device for various proximity aware applications.

The RF portion is a low-cost 2.4GHz BLE Beacon transmitter. With an external 32MHz crystal (XO), it can implement a complete Beacon device. The output power level can be programmed from -10dBm to +5dBm for various applications.

The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that only a minimum of external components is required for a variety of four-electrode LED Bluetooth Beacon transmission AC Body Fat Scales and other related product applications implementation, resulting in reduced component costs and reductions in circuit board areas.

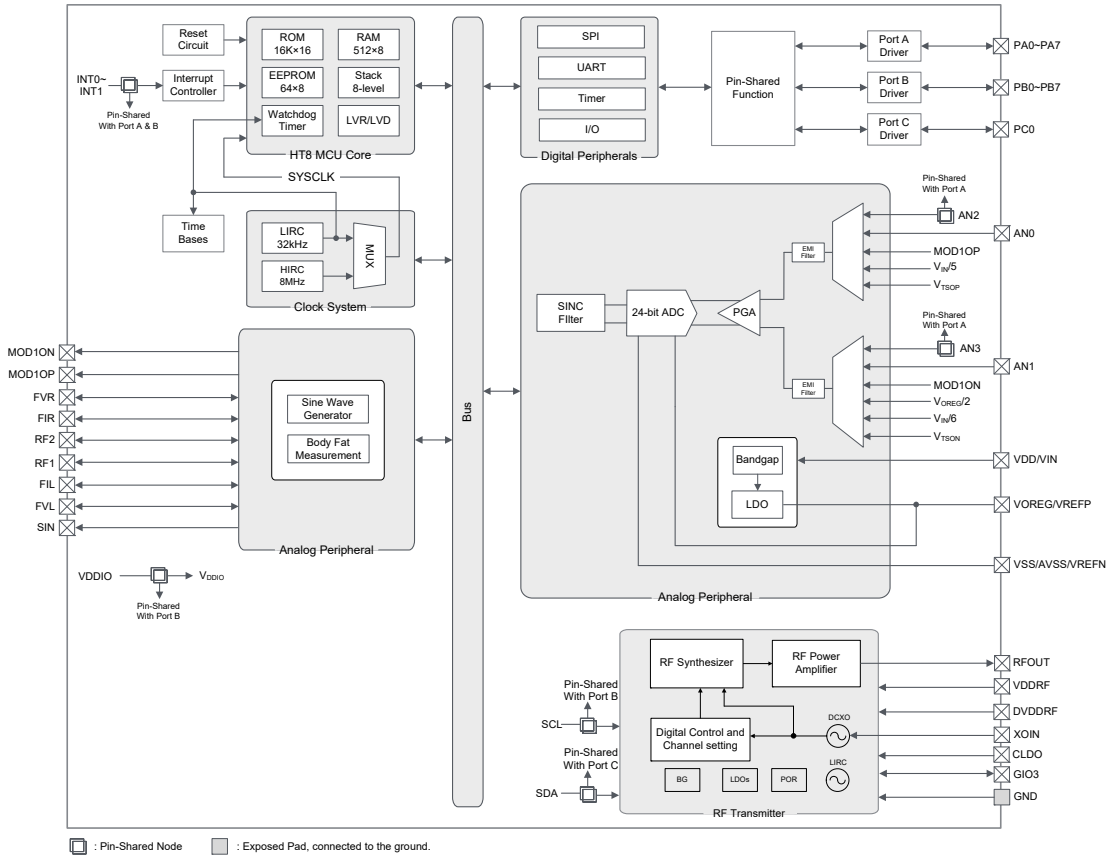
Selection Table

Most features are common to these devices, the main features distinguishing them are Memory capacity and Timer features. The following table summarises the main features of each device.

Device	V_{DD}	Program Memory	Data Memory	Data EEPROM	I/O	A/D
BH66F71652	2.2V~3.6V	8K×16	384×8	32×8	17	24-bit×4
BH66F71662	2.2V~3.6V	16K×16	512×8	64×8	17	24-bit×4

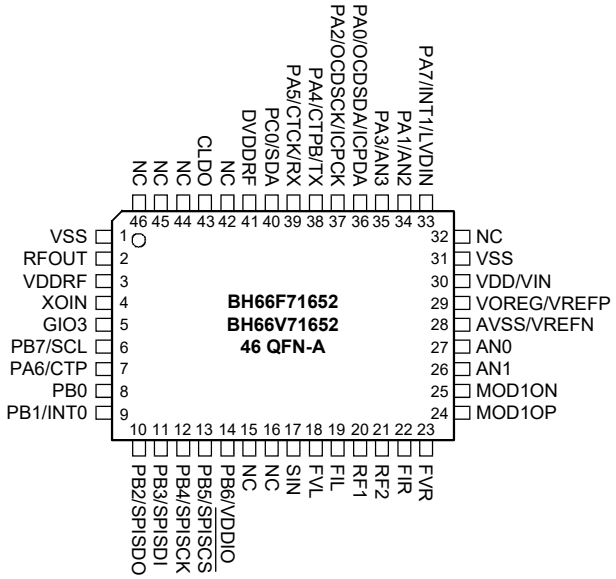
Device	Body Fat Circuit	BLE Beacon			Stack	Package
		Data Rate	Output Power	XO		
BH66F71652	√	1Mbps	-10~+8dBm	32MHz	8	46QFN
BH66F71662	√	1Mbps	-10~+8dBm	32MHz	8	46QFN

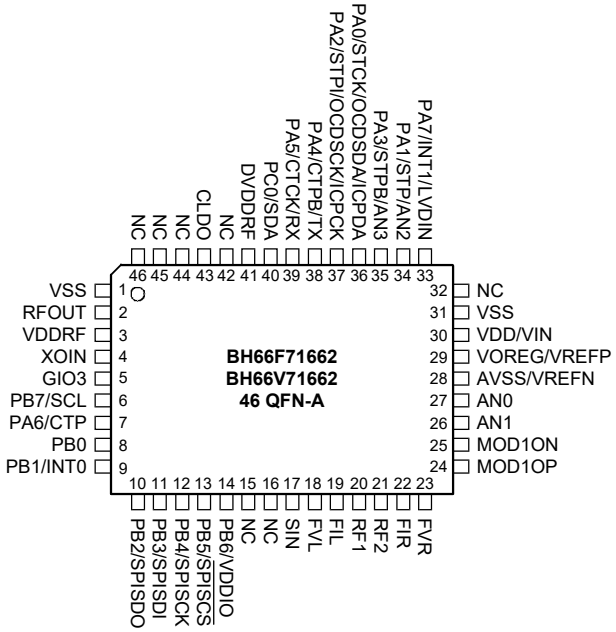
Block Diagram



Note: The figure illustrates the Block Diagram of the device with maximum features, the functional differences between the devices are provided in the Selection Table.

Pin Assignment





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

2. The OCDSDA and OCDSCK pins are supplied as the OCDS dedicated pins and as such only available for the BH66V71652/BH66V71662 devices which are the OCDS EV chips for the BH66F71652/BH66F71662 devices respectively.

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

Pin 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. As the Pin Description table shows the situation for the package with the most pins, not all pins in the table will be available on smaller package sizes. Note that the STM related pins pin-shared with PA0~PA3 are only available for the BH66F71662 device.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/STCK/ OCDSDA/ICPDA	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	STCK	—	ST	—	STM clock input
	OCDSDA	—	ST	CMOS	OCDS address/data pin, for EV chip only
	ICPDA	—	ST	CMOS	ICP address/data
PA1/STP/AN2	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	STP	PAS0	—	CMOS	STM output
	AN2	PAS0	AN	—	A/D Converter external input
PA2/STPI/ OCDSCK/ ICPCK	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	STPI	—	ST	—	STM capture input
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only.
	ICPCK	—	ST	—	ICP clock
PA3/STPB/AN3	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	STPB	PAS0	—	CMOS	STM inverting output
	AN3	PAS0	AN	—	A/D Converter external input
PA4/CTPB/TX	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTPB	PAS1	—	CMOS	CTM inverting output
	TX	PAS1	—	CMOS	UART TX serial data output
PA5/CTCK/RX	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK	PAS1	ST	—	CTM clock input
	RX	PAS1	ST	—	UART RX serial data input
PA6/CTP	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP	PAS1	—	CMOS	CTM output
PA7/INT1/LVDIN	PA7	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT1	PAS1 INTEG INTC0	ST	—	External Interrupt 1 input
	LVDIN	PAS1	AN	—	LVD input
PB0	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
PB1/INT0	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT0	INTEG INTC0	ST	—	External Interrupt 0 input

Pin Name	Function	OPT	I/T	O/T	Description
PB2/SPISDO	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SPISDO	PBS0	—	CMOS	SPI serial data output
PB3/SPISDI	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SPISDI	PBS0	ST	—	SPI serial data input
PB4/SPISCK	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SPISCK	PBS1	ST	CMOS	SPI serial clock
PB5/$\overline{\text{SPISCS}}$	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	$\overline{\text{SPISCS}}$	PBS1	ST	CMOS	SPI slave chip select
PB6/VDDIO	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	VDDIO	PBS1 PMPS	PWR	—	PA6 and PB5~PB0 positive power supply
PB7/SCL	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SCL	—	ST	—	RF I^2C clock line
PC0/SDA	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SDA	—	ST	NMOS	RF I^2C data line
VOREG/VREFP	VOREG	—	—	AN	LDO output
		—	AN	—	Positive power supply for ADC and PGA
	VREFP	—	AN	—	ADC external positive reference input
AVSS/VREFN	AVSS	—	PWR	—	Negative power supply for ADC and PGA
	VREFN	—	AN	—	ADC external negative reference input
AN0~AN1	AN0~AN1	—	AN	—	A/D converter external input
MOD1ON	MOD1ON	—	—	AN	Demodulator negative output
MOD1OP	MOD1OP	—	—	AN	Demodulator positive output
FVR	FVR	—	AN	AN	Right foot channel 1
FIR	FIR	—	AN	AN	Right foot channel 2
RF2	RF2	—	AN	AN	Reference 2 impedance channel
RF1	RF1	—	AN	AN	Reference 1 impedance channel
FIL	FIL	—	AN	AN	Left foot channel 2
FVL	FVL	—	AN	AN	Left foot channel 1
SIN	SIN	—	—	AN	Sine wave output
RFOUT	RFOUT	—	—	AN	RF signal output
XOIN	XOIN	—	AN	—	RF crystal oscillator input
CLDO	CLDO	—	—	AN	LDO output, connected to a bypass capacitor
GIO3	GIO3	—	DI	DO	General purpose pin
VDDRF	VDDRF	—	PWR	—	RF analog positive power supply
DVDDRF	DVDDRF	—	PWR	—	RF digital positive power supply
VDD/VIN	VDD	—	PWR	—	Positive power supply
	VIN	—	PWR	—	LDO input
VSS	VSS	—	PWR	—	Negative power supply

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

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}−0.3V to 6.0V
Input Voltage ..... V_{SS}−0.3V to V_{DD}+0.3V
Storage Temperature..... -50°C to 125°C
Operating Temperature..... -40°C to 85°C
I_{OH} Total..... -80mA
I_{OL} Total 80mA
Total Power Dissipation 500mW
ESD HBM ..... > ±2kV

The device is ESD sensitive. HBM (Human Body Mode) is based on MIL-STD-883.

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

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}=8MHz	2.2	—	3.6	V
	Operating Voltage – LIRC	f_{SYS}=f_{LIRC}=32kHz	2.2	—	3.6	V

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	—	0.2	0.6	0.7	μA
		3V		—	0.2	0.8	1.0	
		2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
	IDLE0 Mode – LIRC	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3.0	5.0	6.0	
	IDLE1 Mode – HIRC	2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	

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

1. Any digital inputs are setup in a non-floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

Operating Current Characteristics

Ta=25°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}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	

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.

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 either 3V.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temperature				
f_{HIRC}	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%	

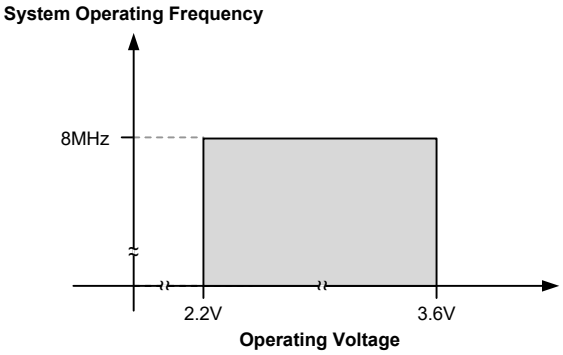
Note: 1. The 3V value for V_{DD} is provided as these is the selectable fixed voltage at which the HIRC frequency is trimmed by the writer.

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

Low Speed Internal Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temperature				
f_{LIRC}	LIRC Frequency	2.2V~3.6V	-40°C~85°C	-10%	32	+10%	kHz
t_{START}	LIRC Start up Time	—	-40°C~85°C	—	—	100	μs

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time Wake-up from 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	42	48	54	ms
	System Reset Delay Time LVRC/WDTC/RSTC 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

Input/Output (without Multi-power) D.C. 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 (except PA6 and PB5~PB0 Pins)	3V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports or Input Pins (except PA6 and PB5~PB0 Pins)	3V	—	$0.8V_{DD}$	—	V_{DD}	V
I_{OL}	Sink Current for I/O Pins (except PA6 and PB5~PB0 Pins)	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
I_{OH}	Source Current for I/O Pins (except PA6 and PB5~PB0 Pins)	3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1,m]=00B (n=0, 1; m=0, 2, 4, 6)	-0.7	-1.5	—	mA
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=01B (n=0, 1; m=0, 2, 4, 6)	-1.3	-2.5	—	mA
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=10B ((n=0, 1; m=0, 2, 4, 6)	-1.8	-3.6	—	mA
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=11B (n=0, 1; m=0, 2, 4, 6)	-4	-8	—	mA
R_{PH}	Pull-high Resistance for I/O Ports (except PA6 and PB5~PB0 Pins) (Note)	3V	—	20	60	100	kΩ
I_{LEAK}	Input Leakage Current (except PA6 and PB5~PB0 Pins)	3V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{TCK}	xTCK Clock Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TPI}	STPI Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	External Interrupt Input Pin Minimum Pulse Width	—	—	10	—	—	μs

Input/Output (with Multi-power) D.C. Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	V_{DD} Power Supply for PA6 and PB5~PB0 Pins	—	—	2.2	—	3.6	V
V_{DDIO}	V_{DDIO} Power Supply for PA6 and PB5~PB0 Pins	—	—	2.2	—	V_{DD}	V
V_{IL}	Input Low Voltage for PA6 and PB5~PB0 Pins	3V	Pin power=V_{DD} or V_{DDIO}, $V_{DDIO}=V_{DD}$	0	—	1.5	V
		—	Pin power=V_{DD} or V_{DDIO}	0	—	0.2 (V_{DD}/V_{DDIO})	
V_{IH}	Input High Voltage for PA6 and PB5~PB0 Pins	3V	Pin power=V_{DD} or V_{DDIO}, $V_{DDIO}=V_{DD}$	3.5	—	5.0	V
		—	Pin power=V_{DD} or V_{DDIO}	0.8 (V_{DD}/V_{DDIO})	—	V_{DD}/V_{DDIO}	
I_{OL}	Sink Current for PA6 and PB5~PB0 Pins	3V	$V_{OL}=0.1(V_{DD}/V_{DDIO})$, $V_{DDIO}=V_{DD}$	16	32	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH}	Source Current for PA6 and PB5~PB0 Pins	3V	V_{OH}=0.9(V_{DD}/V_{DDIO}), V_{DDIO}=V_{DD} SLEDCn[m+1, m]=00B (n=0, 1; m=0, 2, 4, 6)	-0.7	-1.5	—	mA
		3V	V_{OH}=0.9(V_{DD}/V_{DDIO}), V_{DDIO}=V_{DD} SLEDCn[m+1, m]=01B (n=0; m=2, 4 or 6)	-1.3	-2.5	—	mA
		3V	V_{OH}=0.9(V_{DD}/V_{DDIO}), V_{DDIO}=V_{DD} SLEDCn[m+1, m]=10B (n=0; m=2, 4 or 6)	-1.8	-3.6	—	mA
		3V	V_{OH}=0.9(V_{DD}/V_{DDIO}), V_{DDIO}=V_{DD} SLEDCn[m+1, m]=11B (n=0; m=2, 4 or 6)	-4	-8	—	mA
R_{PH}	Pull-high Resistance for PA6 and PB5~PB0 Pins (Note)	3V	Pin power=V_{DD} or V_{DDIO}, V_{DDIO}=V_{DD}	20	60	100	kΩ
I_{LEAK}	Input Leakage Current for PA6 and PB5~PB0 Pins	3V	V_{IN}=V_{SS} or V_{IN}=V_{DD} or V_{DDIO}	—	—	±1	μA

Note: The R_{PH} internal pull-high resistance value is calculated by connecting to ground and enabling 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 / 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	—	Device in SLEEP Mode	1.0	—	—	V

24-bit Delta Sigma A/D Converter Electrical Characteristics

V_{DD}=V_{IN}, Ta=25°C, unless otherwise specified
LDO Test conditions: MCU enters SLEEP mode, other functions disabled

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IN}	LDO Input Voltage	—	—	2.6	—	3.6	V
I_Q	LDO Quiescent Current	—	LDOVS[1:0]=00B, V_{IN}=3.6V, No load	—	400	520	μA
V_{OUT_LDO}	LDO Output Voltage	—	LDOVS[1:0]=00B, V_{IN}=3.6V, I_{LOAD}=0.1mA	-5%	2.4	+5%	V
		—	LDOVS[1:0]=01B, V_{IN}=3.6V, I_{LOAD}=0.1mA		2.6		
		—	LDOVS[1:0]=10B, V_{IN}=3.6V, I_{LOAD}=0.1mA		2.9		
		—	LDOVS[1:0]=11B, V_{IN}=3.6V, I_{LOAD}=0.1mA		3.3		
ΔV_{LOAD}	LDO Load Regulation(1)	—	LDOVS[1:0]=00B, V_{IN}=V_{OUT_LDO} + 0.2V, 0mA ≤ I_{LOAD} ≤10mA	—	0.105	0.210	%/mA
V_{DROP_LDO}	LDO Dropout Voltage(2)	—	LDOVS[1:0]=00B, V_{IN}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}=2%	—	—	220	mV
		—	LDOVS[1:0]=01B, V_{IN}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}=2%	—	—	200	mV
		—	LDOVS[1:0]=10B, V_{IN}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}=2%	—	—	180	mV
		—	LDOVS[1:0]=11B, V_{IN}=3.6V, I_{LOAD}=10mA, ΔV_{OUT_LDO}=2%	—	—	160	mV
TC_{LDO}	LDO Temperature Coefficient	—	Ta=-40°C~85°C, LDOVS[1:0]=00B, V_{IN}=3.6V, I_{LOAD}=100μA	—	—	200	ppm/°C
ΔV_{LINE_LDO}	LDO Line Regulation	—	LDOVS[1:0]=00B, 2.6V ≤ V_{IN} ≤ 3.6V, I_{LOAD}=100μA	—	—	0.2	%/V
ADC & ADC Internal Reference Voltage (Delta Sigma A/D Converter)							
V_{OREG}	Supply Voltage for ADC and PGA	—	LDOEN=0	2.4	—	3.3	V
		—	LDOEN=1	2.4	—	3.3	V
I_{ADC}	Additional Current for ADC Enable	—	—	—	400	550	μA
I_{ADSTB}	Standby Current	—	MCU enters SLEEP mode, no load	—	—	1	μA
N_R	Resolution	—	—	—	—	24	Bit
INL	Integral Nonlinearity	—	V_{OREG}=3.3V, V_{REF}=1.25V, ΔSI=±450mV, PGA gain=1	—	±50	±200	ppm
NFB	Noise Free Bits	—	V_{REF}=1.2V, Gain=32, Data rate=8Hz	—	16.8	—	Bit
			V_{REF}=1.2V, Gain=64, Data rate=8Hz	—	15.9	—	Bit
			V_{REF}=1.2V, Gain=128, Data rate=8Hz	—	15.0	—	Bit
ENOB	Effective Number of Bits	—	V_{REF}=1.2V, Gain=32, Data rate=8Hz	—	19.5	—	Bit
			V_{REF}=1.2V, Gain=64, Data rate=8Hz	—	18.6	—	Bit
			V_{REF}=1.2V, Gain=128, Data rate=8Hz	—	17.7	—	Bit
f_{ADCK}	ADC Clock Frequency	—	—	40	409.6	440	kHz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{ADO}	ADC Output Data Rate	—	f_{MCLK}=4MHz, FLMS[2:0]=000B	4	—	521	Hz
		—	f_{MCLK}=4MHz, FLMS[2:0]=010B	10	—	1302	Hz
V_{REFP}	External Reference Input Voltage	—	—	V_{REFN} +0.8	—	V_{OREG}	V
V_{REFN}		—		0	—	V_{REFP} -0.8	V
V_{REF}		—	$V_{REF}=(V_{REFP}–V_{REFN})×VREFGN$	0.8	—	1.75	V
PGA							
V_{CM_PGA}	Common Mode Voltage Range	—	—	0.4	—	V_{OREG} -0.95	V
ΔD_I	Differential Input Voltage Range	—	Gain=PGAGN×ADGN	-V_{REF} /Gain	—	+V_{REF} /Gain	V
Temperature Sensor							
TC_{TS}	Temperature Sensor Temperature Coefficient	—	Ta=-40°C~85°C, VGS[1:0]=00B (Gain=1)	—	175	—	μV/°C

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

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

Effective Number of Bits (ENOB)

V_{REF}=1.2V, f_{ADCK}=133kHz

Data Rate (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
4	21.2	21.2	21.2	21.0	20.6	20.1	19.2	18.1
8	20.7	20.7	20.6	20.6	20.2	19.5	18.6	17.7
16	20.2	20.1	20.2	20.0	19.7	19.0	18.2	17.2
33	19.5	19.5	19.5	19.5	19.2	18.6	17.7	16.7
65	19.2	19.2	19.1	18.9	18.6	18.0	17.0	16.1
130	18.4	18.4	18.4	18.2	17.9	17.3	16.5	15.5
260	16.9	16.8	16.8	16.8	16.7	16.5	15.9	15.0
521	15.0	15.0	14.9	14.9	14.9	14.9	14.7	14.3

V_{REF}=1.2V, f_{ADCK}=333kHz

Data Rate (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
10	21.2	21.1	21.0	20.8	20.2	19.4	18.6	17.6
20	20.7	20.7	20.6	20.3	19.8	19.0	18.0	17.0
41	20.2	20.1	20.0	19.8	19.3	18.5	17.6	16.6
81	19.6	19.5	19.5	19.2	18.7	17.9	17.1	16.0
163	19.2	19.1	19.0	18.8	18.2	17.4	16.5	15.5
326	18.4	18.4	18.3	18.0	17.7	16.8	16.0	15.0
651	16.8	16.8	16.8	16.8	16.6	16.2	15.5	14.6
1302	15.0	14.9	14.9	14.9	14.9	14.8	14.6	14.1

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 2.1V	-5%	2.1	+5%	V
		—	LVR enable, voltage select 2.55V		2.55		
		—	LVR enable, voltage select 3.15V		3.15		
		—	LVR enable, voltage select 3.8V		3.8		
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 1.04V	-10%	1.04	+10%	V
		—	LVD enable, voltage select 2.2V	-5%	2.2	+5%	
		—	LVD enable, voltage select 2.4V		2.4		
		—	LVD enable, voltage select 2.7V		2.7		
		—	LVD enable, voltage select 3.0V		3.0		
		—	LVD enable, voltage select 3.3V		3.3		
		—	LVD enable, voltage select 3.6V		3.6		
		—	LVD enable, voltage select 4.0V		4.0		
$I_{LVRLVDBG}$	Operating Current	3V	LVD enable, LVR enable, VBGEN=0	—	—	18	μA
		5V		—	20	25	
		3V	LVD enable, LVR enable, VBGEN=1	—	—	150	μA
		5V		—	180	200	
t_{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	15	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs
I_{LVR}	Additional Current for LVR Enable	—	LVD disable, VBGEN=0	—	—	24	μA

RF Electrical Characteristics

Ta=25°C, V_{DD}=3.3V, f_{XTAL}=32MHz,
GFSK modulation with matching circuit, unless otherwise specified

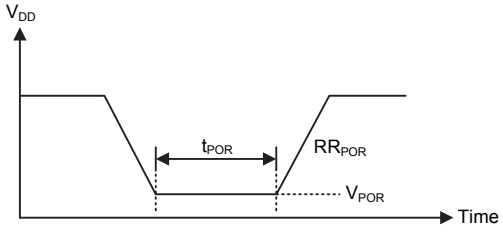
Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_{OP}	Operating Temperature	—	-40	—	85	°C
V_{DD}	Supply Voltage	—	2.2	3.3	3.6	V
Digital I/Os						
V_{IH}	High Level Input Voltage	—	0.7×V_{DD}	—	V_{DD}	V
V_{IL}	Low Level Input Voltage	—	0	—	0.3×V_{DD}	V
V_{OH}	High Level Output Voltage	I_{OH}=-5mA	0.8×V_{DD}	—	V_{DD}	V
V_{OL}	Low Level Output Voltage	I_{OL}=5mA	0	—	0.2×V_{DD}	V
Current Consumption						
$I_{DeepSleep}$	Deep Sleep Mode Current	—	—	0.35	1.00	μA
$I_{LightSleep}$	Light Sleep Mode Current	X'tal On	—	1	—	mA
I_{TX}	TX Mode Current (High Power Matching)	RF output power=-10dBm	—	10.5	—	mA
		RF output power=-5dBm	—	12.5	—	
		RF output power=-2dBm(Note)	—	15.5	—	
		RF output power=5dBm	—	19	—	
		RF output power=8dBm	—	24	—	

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Transmitter Characteristics						
f_{RF}	RF Operating Frequency	CH37	—	2402	—	MHz
		CH38	—	2426	—	
		CH39	—	2480	—	
DR	Data Rate	GFSK@f_{DEV}=250kHz	—	1	—	Mbps
f_{DEV}	Frequency Deviation	—	—	250	—	kHz
P_{OUT}	RF Transmit Output Power	—	-10	-2	8	dBm
t_{ST}	RF Transmit Settling Time	Light sleep mode to Transmit mode		TBD		
$f_{channel}$	Channel Spacing	Non-overlapping channel spacing	—	2	—	MHz
	Occupied Bandwidth	—	—	1	—	MHz
SE_{TX}	TX Spurious Emission (P_{OUT}=5dBm)	f < 1GHz	—	—	-36	dBm
		47MHz < f < 74MHz	—	—	-54	
		87.5MHz < f < 108MHz				
		174MHz < f < 230MHz				
		470MHz < f < 862MHz				
		f > 1GHz	—	—	-30	
	Adjacent Channel Power	f_{RF}±2MHz	—	—	-20	dBm
		f_{RF}±(3+n)MHz; where n=0, 1, 2…	—	—	-30	
LO Characteristics						
F_{LO}	Frequency Coverage Range	—	2400	—	2500	MHz
f_{STEP}	Frequency Synthesizer Step	—	—	—	10	kHz
PN	2.4GHz Phase Noise	PN@100k offset	—	-85	—	dBc/Hz
		PN@1M offset	—	-105	—	
Crystal (X'tal) Oscillator						
f_{XTAL}	X'tal Frequency	General case	—	32	—	MHz
ESR	X'tal Equivalent Series Resistance	—	—	—	100	Ω
C_{LOAD}	X'tal Capacitor Load	—	12	—	16	pF
	X'tal Tolerance	—	-20	—	+20	ppm
$t_{Startup}$	X'tal Settling Time	49US with a 12pF C_{LOAD}	—	—	0.8	ms
		3225 SMD with a 12pF C_{LOAD}	—	—	1	ms

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

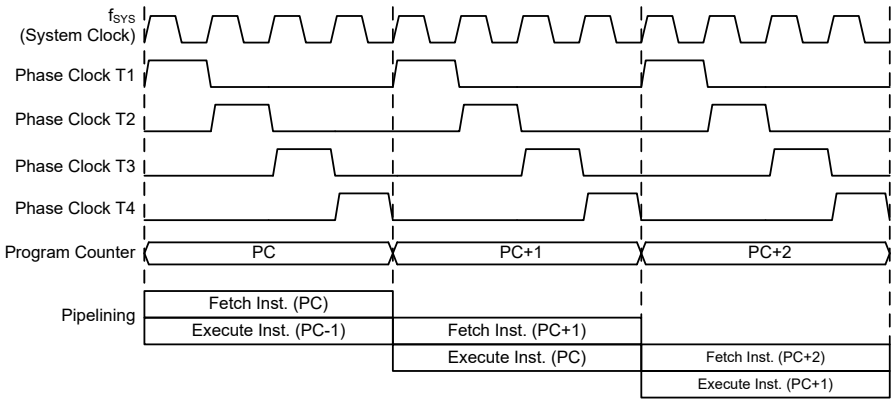


System Architecture

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

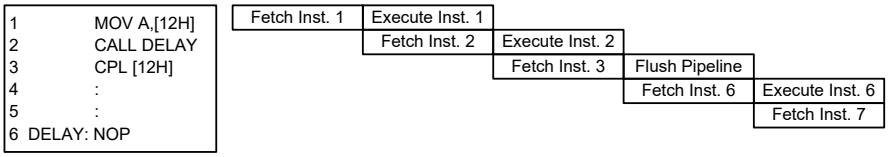
Clocking and Pipelining

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.



System Clocking and Pipelining

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



Instruction Fetching

Program Counter

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

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

Device	Program Counter	
	Program Counter High Byte	PCL Register
BH66F71652	PC12~PC8	PCL7~PCL0
BH66F71662	PBP0, PC12~PC8	PCL7~PCL0

Program Counter

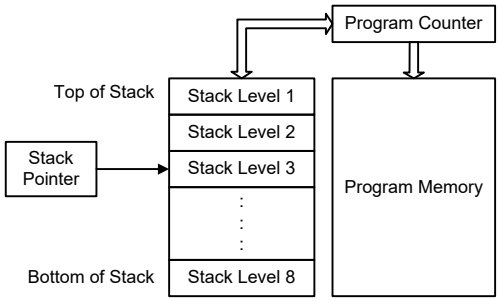
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 8 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



Arithmetic and Logic Unit – ALU

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

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

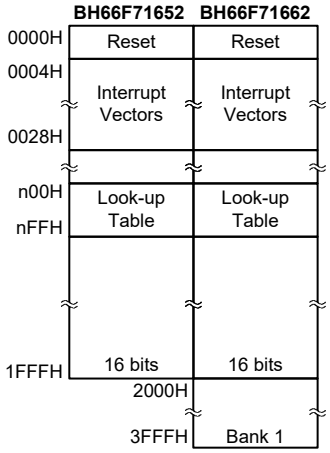
Flash Program Memory

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

Device	Capacity	Banks
BH66F71652	8K×16	0
BH66F71662	16K×16	0~1

Structure

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



Program Memory Structure

Special Vectors

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

Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the corresponding table read instruction such as “TABRD [m]” or “TABRDL [m]” respectively when the memory [m] is located in Sector 0. If the memory [m] is located in other sectors except Sector 0, the data can be retrieved from the program memory using the corresponding extended table read instruction such as “LTABRD [m]” or “LTABRDL [m]” respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user

defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as "0".

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

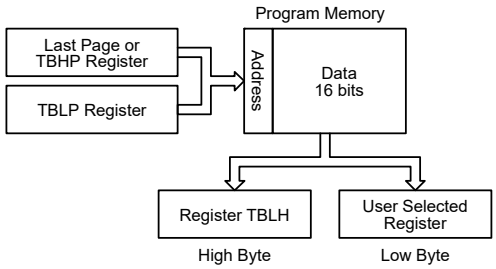


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "1F00H" which refers to the start address of the last page within the 8K words Program Memory of the BH66F71652 device. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "1F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by the TBLP and TBHP registers if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

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

Table Read Program Example

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

```
    :
    :
code1 .section 'code'
org 1F00h           ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

In Circuit Programming – ICP

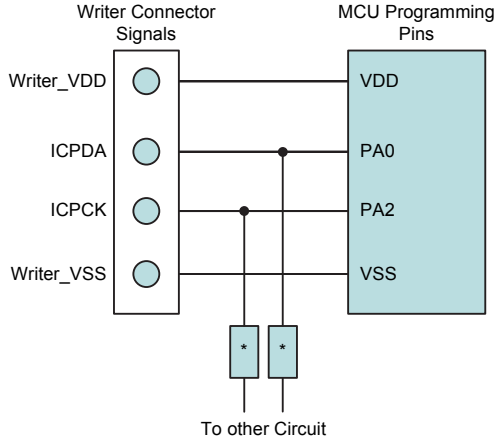
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

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 devices 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 BH66V716x2 which is used to emulate the real MCU device named BH66F716x2. The EV chip device also provides the “On-Chip Debug” function to debug the real MCU device during development process. The EV chip and real MCU device, BH66V716x2 and BH66F716x2, are almost functional compatible except the “On-Chip Debug” function. Users can

use the EV chip device to emulate the real MCU device behaviors 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, the corresponding pin functions shared with the OCDSDA and OCDSCK pins in the real MCU 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 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 devices. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.

There is another area of the Data Memory reserved for the Body Fat Sine Wave Pattern. The addresses of the Sine Wave Pattern Memory area overlop those in the Special Purpose Data Memory area.

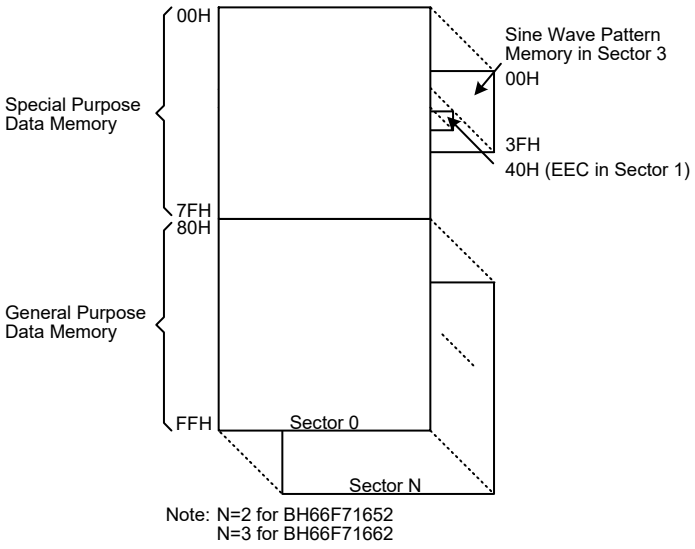
Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value if using the indirect addressing method.

Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide RAM. Each of the Data Memory Sector is categorized into two types, the special Purpose Data Memory and the General Purpose Data Memory. While the 00H~3FH of Sector 3 is Sine Pattern Memory. The address range of the Special Purpose Data Memory for the devices is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH.

Device	Special Purpose Data Memory	Sine Pattern Data Memory		General Purpose Data Memory	
	Located Sectors	Capacity	Sector: Address	Capacity	Sector: Address
BH66F71652	0, 1	64×8	3: 00H~3FH	384×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH
BH66F71662	0, 1	64×8	3: 00H~3FH	512×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH

Data Memory Summary



Data Memory Structure

Data Memory Addressing

For the devices that support the extended instructions, there is no Bank Pointer for Data Memory. The Bank Pointer, PBP, is only available for the Program Memory of the BH66F71662 device. For Data Memory the desired Sector is pointed by the MP1H or MP2H register and the certain Data Memory address in the selected sector is specified by the MP1L or MP2L register when using indirect addressing access.

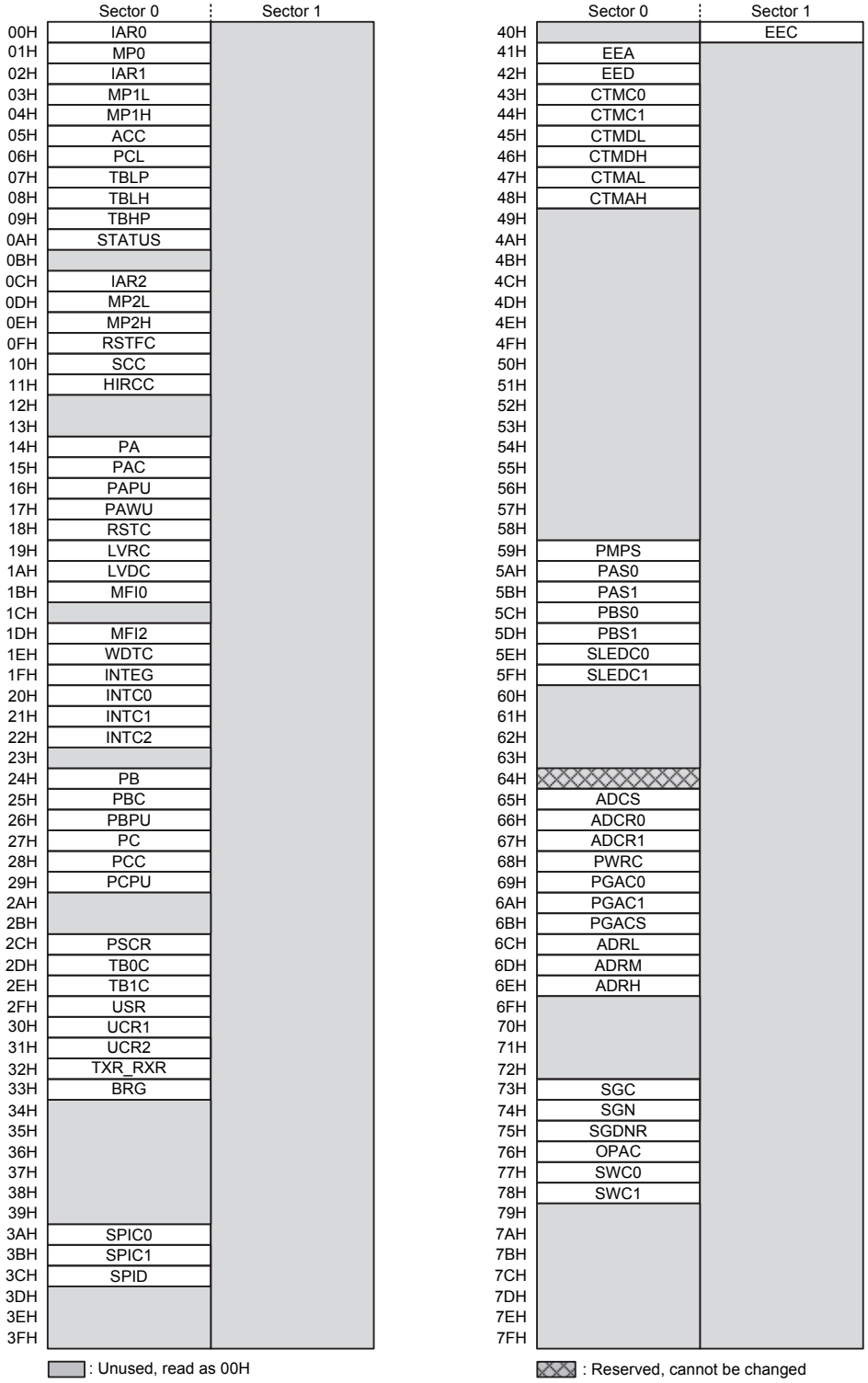
Direct Addressing can be used in all sectors using the extended instructions which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the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

General Purpose Data Memory

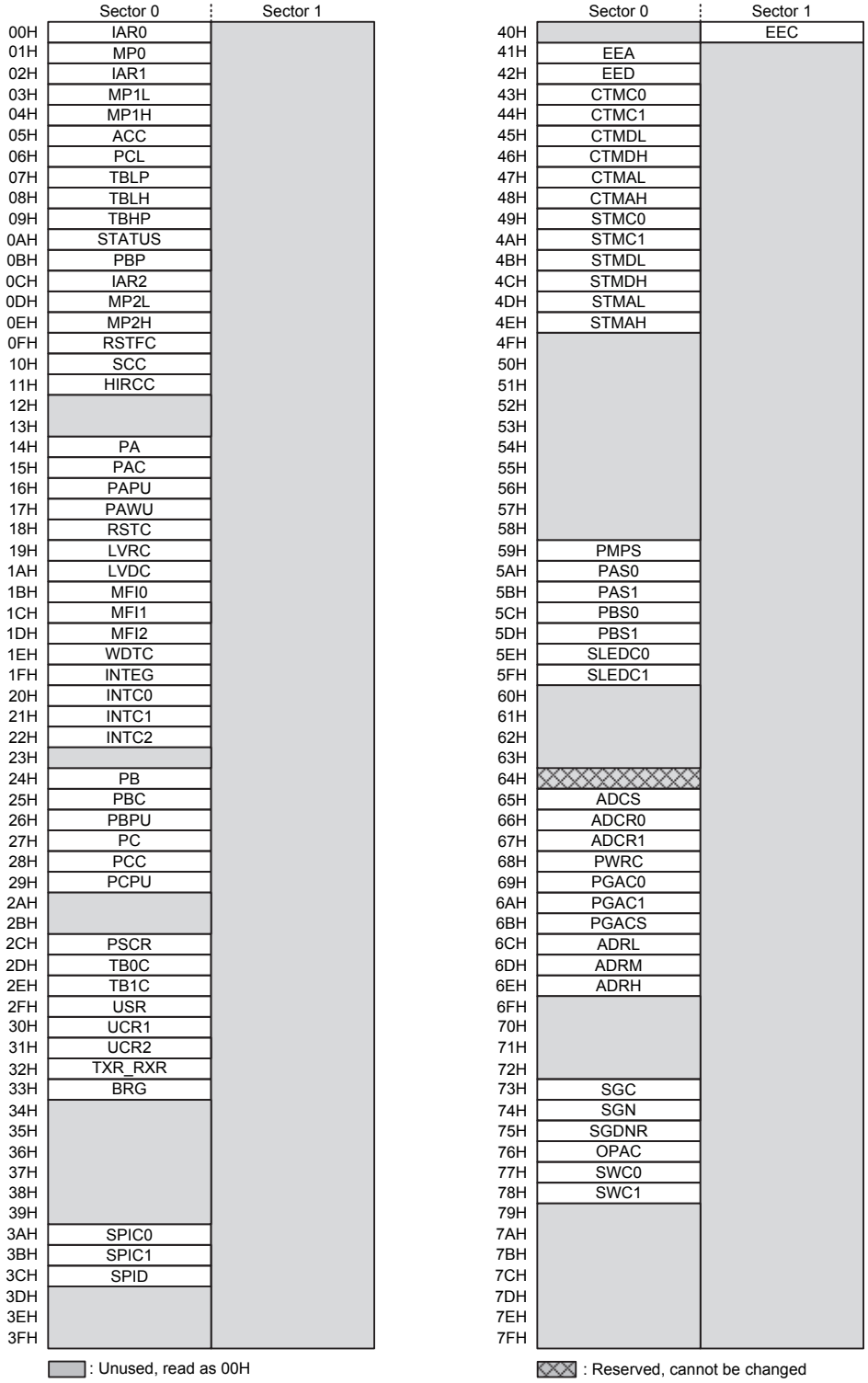
All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user programing for both reading and writing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

Special Purpose Data Memory

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



Special Purpose Data Memory – BH66F71652



Special Purpose Data Memory – BH66F71662

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, IAR2

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

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

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

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

Indirect Addressing Program Example 1

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

Indirect Addressing Program Example 2

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

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

Direct Addressing Program Example using extended instructions

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

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

Program Memory Bank Pointer – PBP

For the BH66F71662 device the Program Memory is divided into two banks. Selecting the required Program Memory area is achieved using the Program Memory Bank Pointer, PBP. The PBP register should be properly configured before the device executes the "Branch" operation using the "JMP" or "CALL" instruction. After that a jump to a non-consecutive Program Memory address which is located in a certain bank selected by the program memory bank pointer bits will occur.

• **PBP Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **PBP0**: Program 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 SC flag, CZ flag, zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

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

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

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

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

• STATUS Register

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

"x": unknown

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

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

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

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

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

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

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

Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

The "C" flag is also affected by a rotate through carry instruction.

EEPROM Data Memory

The devices contain 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.

Device	Capacity	Address
BH66F71652	32×8	00H~1FH
BH66F71662	64×8	00H~3FH

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 32×8~64×8 bits for these devices. 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 Sector 0 and a single control register in Sector 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 Sector 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Sector 1, can only be read from or written to indirectly using the MP1L/MP1H or MP2L/MP2H Memory Pointer pairs and Indirect Addressing Register, IAR1/IAR2. Because the EEC control register is located at address 40H in Sector 1, the MP1L or MP2L Memory Pointer must first be set to the value 40H and the MP1H or MP2H Memory Pointer high byte 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 (BH66F71652)	—	—	—	EEA4	EEA3	EEA2	EEA1	EEA0
EEA (BH66F71662)	—	—	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM Register List

- **EEA Register – BH66F71652**

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

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **EEA4~EEA0**: Data EEPROM address bit 4 ~ bit 0

- **EEA Register – BH66F71662**

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	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
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 **EED7~EED0**: 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 EEC register content.

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. Then the read enable bit, RDEN, in the EEC register must 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. 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 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 Memory Pointer high byte register, MP1H or MP2H, will be reset to zero, which means that Data Memory Sector 0 will be selected. As the EEPROM control register is located in Sector 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

EEPROM Interrupt

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

Programming Considerations

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

When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. 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, 40H                   ; setup memory pointer low byte MP1L
MOV  MP1L, A                  ; MP1L points to EEC register
MOV  A, 01H                   ; setup memory pointer high byte MP1H
MOV  MP1H, 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  MP1H
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, 40H                   ; setup memory pointer low byte MP1L
MOV  MP1L, A                  ; MP1L points to EEC register
MOV  A, 01H                   ; setup memory pointer high byte MP1H
MOV  MP1H, 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  MP1H
```

Oscillators

Various oscillator options offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving. Oscillator selections and operation are selected through the application program 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. All oscillator options are selected through the relevant control registers. 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 oscillators. With the capability of dynamically switching between fast and slow system clock, the devices have the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

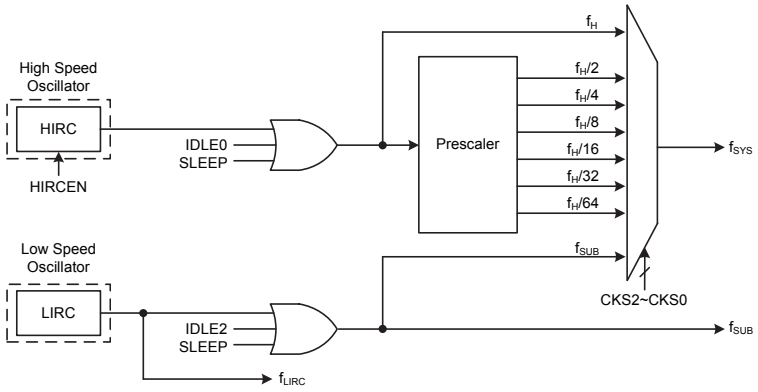
Type	Name	Frequency
Internal High Speed RC	HIRC	8MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two oscillator sources, a high speed oscillator and a low speed oscillator. The high speed oscillator is the internal 8MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC.

The frequency of the slow speed or high speed system clock is also determined using the CKS2~CKS0 bits in the SCC register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators.



System Clock Configurations

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Internal 32kHz Oscillator – LIRC

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

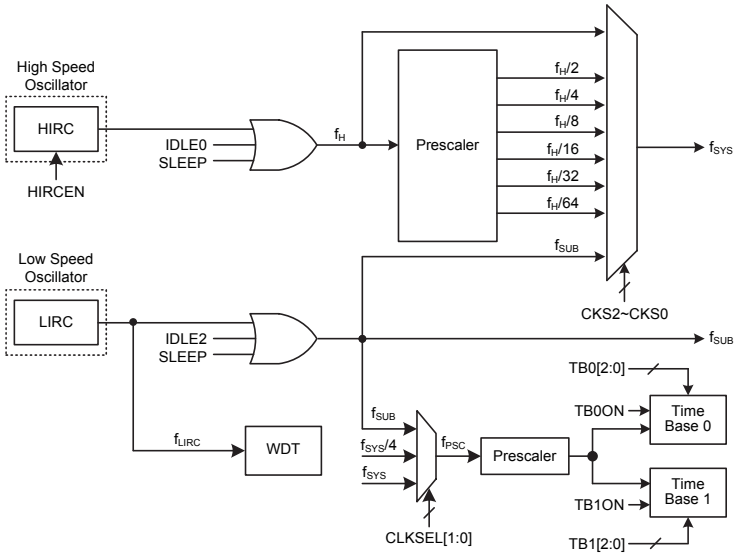
Operating Modes and System Clocks

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

System Clocks

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

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



Device Clock Configurations

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

System Operation Modes

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

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

"x": Don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.
2. The f_{LIRC} clock can be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

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 HIRC oscillator. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio

being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

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

SLEEP Mode

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

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

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

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

System Operating Mode Control Register List

- **SCC Register**

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

Bit 7~5 **CKS2~CKS0**: System clock selection
000: f_H
001: $f_H/2$
010: $f_H/4$

011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: f_{SUB}

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

Bit 4~2 Unimplemented, read as "0"

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

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

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

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

• HIRCC Register

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

Bit 7~2 Unimplemented, read as "0"

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

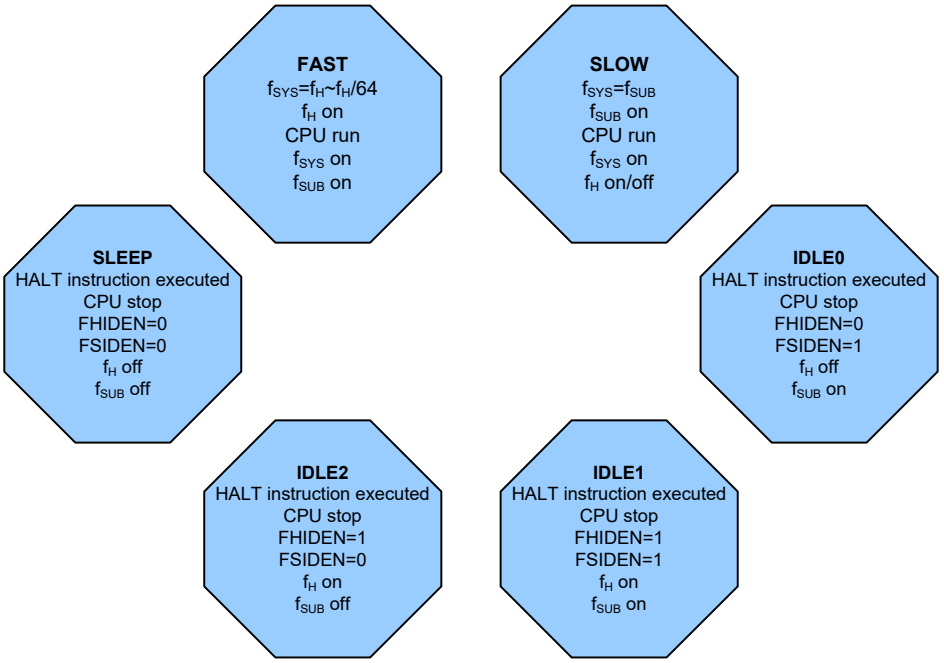
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set high to enable the HIRC oscillator, the HIRCF bit will first be cleared to zero and then set high after the HIRC oscillator is stable.

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

Operating Mode Switching

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

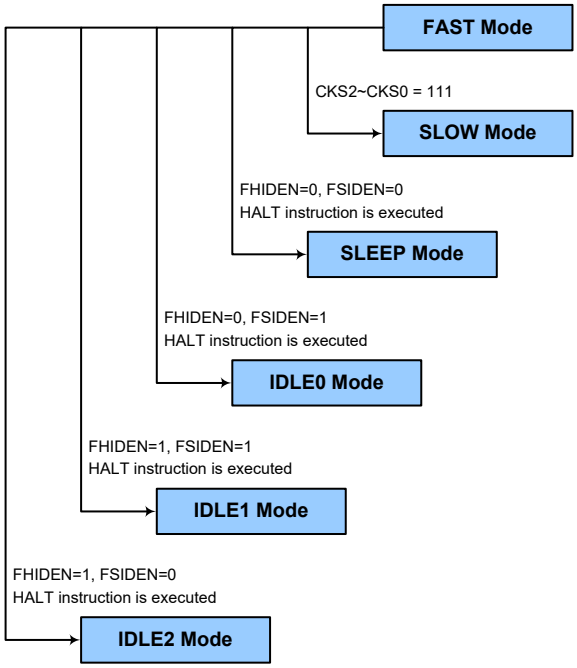
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE 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 set the CKS2~CKS0 bits to "111" in the SCC register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

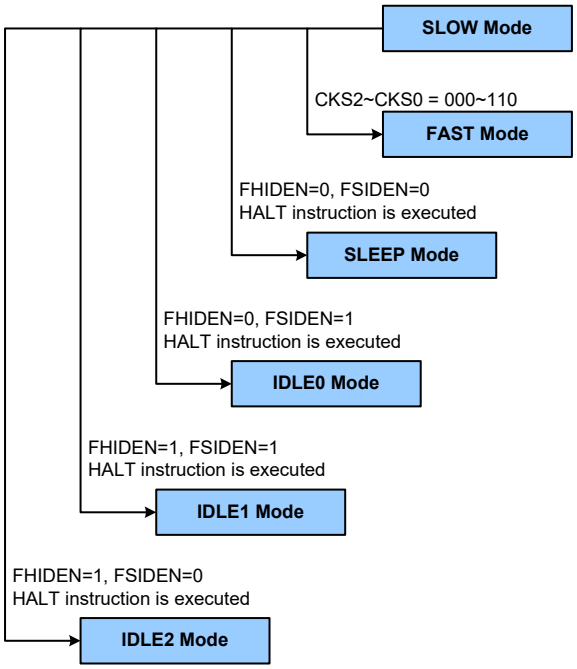
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

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

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

Entering the IDLE2 Mode

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

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

Standby Current Considerations

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

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

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

Wake-up

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

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

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

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

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

Watchdog Timer

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

Watchdog Timer Clock Source

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

Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period as well as the enable/disable and 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 software control
10101: Disable
01010: Enable
Other values: Reset MCU
When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time,t_{SRESET} and the WRF bit in the RSTFC register will be set high.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

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

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

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

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

Watchdog Timer Operation

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

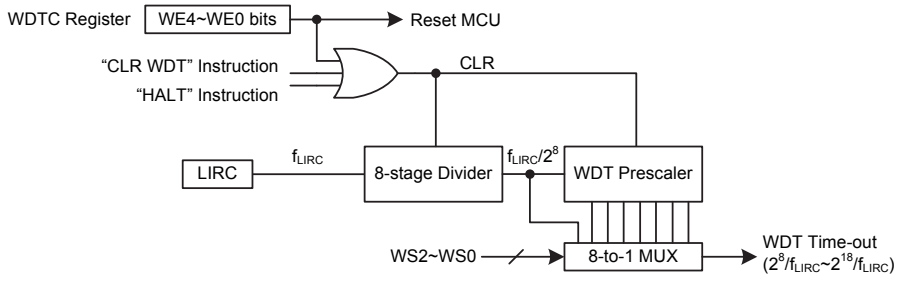
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

Watchdog Timer Enable/Disable Control

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

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

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



Watchdog Timer

Reset and Initialisation

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

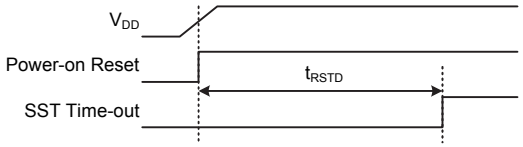
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

Reset Functions

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

Power-on Reset

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



Power-on Reset Timing Chart

Internal Reset Control

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

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

Internal Reset Function Control

• **RSTC Register**

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

Bit 7~0 **RSTC7~RSTC0**: Reset function control
01010101: No operation
10101010: No operation
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET} and the RSTF bit in the RSTFC register will be set to 1.

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

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

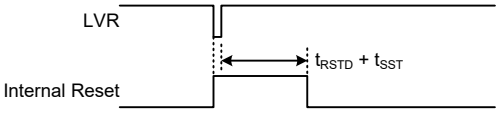
Bit 2 **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.

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

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

Low Voltage Reset – LVR

The microcontrollers contain a low voltage reset circuit in order to monitor the supply voltage of the device and provide an MCU reset should the value fall below a certain predefined level. The LVR function is always enabled in FAST and SLOW Mode with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVR/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 can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.



Low Voltage Reset Timing Chart

• **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage Select control
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Any other value: Generates MCU reset – register is reset to POR value

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

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

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

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

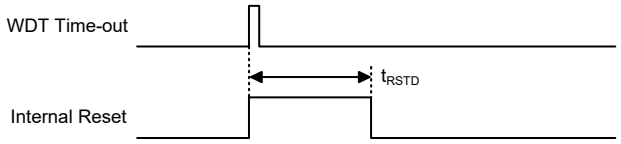
Bit 2 **LVRF**: LVR function reset flag
0: Not occur
1: Occurred
This bit is set high when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

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

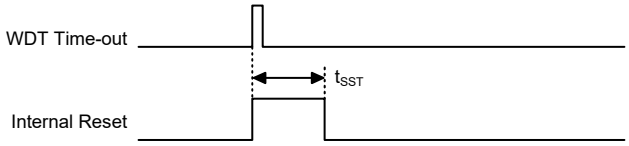
The Watchdog time-out flag TO will be set to "1" when Watchdog time-out Reset during normal operation.



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	LVR reset during 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" stands for unchanged

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

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

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers. Note that where more than one package type exists the table reflects the situation for the larger package type.

Register	BH66F71652	BH66F71662	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	●	●	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	●		---x xxxx	---u uuuu	---u uuuu	---u uuuu
		●	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	●	●	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
PBP		●	---- ---0	---- ---0	---- ---0	---- ---u
IAR2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	●	●	---- 0x00	---- u1uu	---- uuuu	---- uuuu
SCC	●	●	000- --00	000- --00	000- --00	uuu- --uu
HIRCC	●	●	---- --01	---- --01	---- --01	---- --uu
PA	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTC	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
LVRC	●	●	0101 0101	uuuu uuuu	0101 0101	uuuu uuuu
LVDC	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
MFI0	●	●	--00 --00	--00 --00	--00 --00	--uu --uu
MFI1		●	--00 --00	--00 --00	--00 --00	--uu --uu
MFI2	●	●	--00 --00	--00 --00	--00 --00	--uu --uu
WDTC	●	●	0101 0011	0101 0011	0101 0011	uuuu uuuu
INTEG	●	●	---- 0000	---- 0000	---- 0000	---- uuuu
INTC0	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	●		00-0 00-0	00-0 00-0	00-0 00-0	uu-u uu-u
		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	●	●	-000 -000	-000 -000	-000 -000	-uuu -uuu
PB	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	●	●	---- ---1	---- ---1	---- ---1	---- ---u
PCC	●	●	---- ---1	---- ---1	---- ---1	---- ---u
PCPU	●	●	---- ---0	---- ---0	---- ---0	---- ---u
PSCR	●	●	---- --00	---- --00	---- --00	---- --uu
TB0C	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
TB1C	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
USR	●	●	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	●	●	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	BH66F71652	BH66F71662	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TXR_ RXR	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SPIC0	●	●	111- --00	111- --00	111- --00	uuu- --uu
SPIC1	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
SPID	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
EEA	●		---0 0000	---0 0000	---0 0000	---u uuuu
		●	--00 0000	--00 0000	--00 0000	--uu uuuu
EED	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMC0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDH	●	●	---- --00	---- --00	---- --00	---- --uu
CTMAL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMAH	●	●	---- --00	---- --00	---- --00	---- --uu
STMC0		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMC1		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDH		●	---- --00	---- --00	---- --00	---- --uu
STMAL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAH		●	---- --00	---- --00	---- --00	---- --uu
PMPS	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
PAS0	●	●	00-- 00--	00-- 00--	00-- 00--	uu-- uu--
PAS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	●	●	0000 ----	0000 ----	0000 ----	uuuu ----
PBS1	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
SLEDC0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC1	●	●	---- --00	---- --00	---- --00	---- --uu
ADCS	●	●	---0 0000	---0 0000	---0 0000	---u uuuu
ADCR0	●	●	0010 00--	0010 00--	0010 00--	uuuu uu--
ADCR1	●	●	000- -00-	000- -00-	000- -00-	uuu- -uu-
PWRC	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
PGAC0	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
PGAC1	●	●	-000 000-	-000 000-	-000 000-	-uuu uuu-
PGACS	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADRL	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRM	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SGC	●	●	0000 ----	0000 ----	0000 ----	uuuu ----
SGN	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
SGDNR	●	●	---0 0000	---0 0000	---0 0000	---u uuuu
OPAC	●	●	0--- 0000	0--- 0000	0--- 0000	u--- uuuu
SWC0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SWC1	●	●	---- --00	---- --00	---- --00	---- --uu
EEC	●	●	---- 0000	---- 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 devices provide bidirectional input/output lines labeled with port names PA~PC. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A, [m]", where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	—	—	—	—	—	—	PC0
PCC	—	—	—	—	—	—	—	PCC0
PCPU	—	—	—	—	—	—	—	PCPU0

"—": Unimplemented, read as "0"

I/O Logic Function Register List

Pull-high Resistors

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

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

- **PxPU Register**

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

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

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

As the PB7 and PC0 lines are internally connected with the RF transmitter, the PBPU7 bit in the PBPU register should be fixed at "0" after power on.

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-shared functional pin is selected as general purpose input and the MCU enters the IDLE or SLEEP mode.

• **PAWU Register**

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

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

I/O Port Control Registers

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

• **PxC Register**

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

PxCn: I/O Port x pin type selection
0: Output
1: Input

The PxCn bit is used to control the pin type selection. Here the “x” can be A, B and C. However, the actual available bits for each I/O Port may be different.

It is important to note that the PBC7 bit in the PBC register must be cleared to "0" to setup the corresponding line as output after power on. This will ensure correct internal connection between the MCU and RF transmitter.In addition, the PCC0 bit in the PCC register must be properly configured if the RF pin function shared with PC0 is to be used. Setting the PC0 as input or output depends on which of the RF pin function to be used as input or output.

I/O Port Source Current Selection

The source current of each pin in the devices can be configured with different source current which is selected by the corresponding pin source current select bits. These source current bits are available when the corresponding pin is configured as a CMOS output. Otherwise, these select bits have no effect. Users should refer to the Input/Output Characteristics section to obtain the exact value for different applications.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLEDC0	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
SLEDC1	—	—	—	—	—	—	SLEDC11	SLEDC10

I/O Port Source Current Selection Register List

• SLEDC0 Register

Bit	7	6	5	4	3	2	1	0
Name	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
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 **SLEDC07~SLEDC06**: PB7~PB4 source current selection
00: Level 0 (min.)
01: Level 1
10: Level 2
11: Level 3 (max.)

Bit 5~4 **SLEDC05~SLEDC04**: PB3~PB0 source current selection
00: Level 0 (min.)
01: Level 1
10: Level 2
11: Level 3 (max.)

Bit 3~2 **SLEDC03~SLEDC02**: PA7~PA4 source current selection
00: Level 0 (min.)
01: Level 1
10: Level 2
11: Level 3 (max.)

Bit 1~0 **SLEDC01~SLEDC00**: PA3~PA0 source current selection
00: Level 0 (min.)
01: Level 1
10: Level 2
11: Level 3 (max.)

• SLEDC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	SLEDC11	SLEDC10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **SLEDC11~SLEDC10**: PC0 source current selection
00: Level 0 (min.)
01: Level 1
10: Level 2
11: Level 3 (max.)

I/O Port Power Source Control

These devices support different I/O port power source selections for PA6 and PB0~PB5. The port power can come from either the power pin VDD or VDDIO which is determined using the PMPS bit field in the PMPS register. The VDDIO power pin function should first be selected using the corresponding pin-shared function selection bits if the port power is supposed to come from the VDDIO pin. An important point to know is that the input power voltage on the VDDIO pin should be equal to or less than the device supply power voltage when the VDDIO pin is selected as the port power supply pin.

- **PMPS Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PMPS5~PMPS4**: PB5~PB4 pin power source selection
0x: VDD
1x: VDDIO

Bit 3~2 **PMPS3~PMPS2**: PB3~PB0 pin power source selection
0x: VDD
1x: VDDIO

Bit 1~0 **PMPS1~PMPS0**: PA6 pin power source selection
0x: VDD
1x: VDDIO

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as INT1, CTCK etc, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bit fields. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be setup as an input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	—	—	PAS03	PAS02	—	—
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	—	—	—	—
PBS1	—	—	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10

Pin-shared Function Selection Register List

• PAS0 Register – BH66F71652

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	—	—	PAS03	PAS02	—	—
R/W	R/W	R/W	—	—	R/W	R/W	—	—
POR	0	0	—	—	0	0	—	—

Bit 7~6 **PAS07~PAS06**: PA3 pin-shared function selection
00: PA3
01: PA3
10: AN3
11: PA3

Bit 5~4 Unimplemented, read as "0"

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00: PA1
01: PA1
10: AN2
11: PA1

Bit 1~0 Unimplemented, read as "0"

• PAS0 Register – BH66F71662

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	—	—	PAS03	PAS02	—	—
R/W	R/W	R/W	—	—	R/W	R/W	—	—
POR	0	0	—	—	0	0	—	—

Bit 7~6 **PAS07~PAS06**: PA3 pin-shared function selection
00: PA3
01: STPB
10: AN3
11: PA3

Bit 5~4 Unimplemented, read as "0"

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00: PA1
01: STP
10: AN2
11: PA1

Bit 1~0 Unimplemented, read as "0"

• **PAS1 Register**

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

Bit 7~6 **PAS17~PAS16**: PA7 pin-shared function selection
00: PA7/INT1
01: LVDIN
10: PA7/INT1
11: PA7/INT1

Bit 5~4 **PAS15~PAS14**: PA6 pin-shared function selection
00: PA6
01: CTP
10: PA6
11: PA6

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

Bit 1~0 **PAS11~PAS10**: PA4 pin-shared function selection
00: PA4
01: CTPB
10: TX
11: PA4

• **PBS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

Bit 7~6 **PBS07~PBS06**: PB3 pin-shared function selection
00: PB3
01: SPISDI
10: PB3
11: PB3

Bit 5~4 **PBS05~PBS04**: PB2 pin-shared function selection
00: PB2
01: SPISDO
10: PB2
11: PB2

Bit 3~0 Unimplemented, read as “0”

• **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

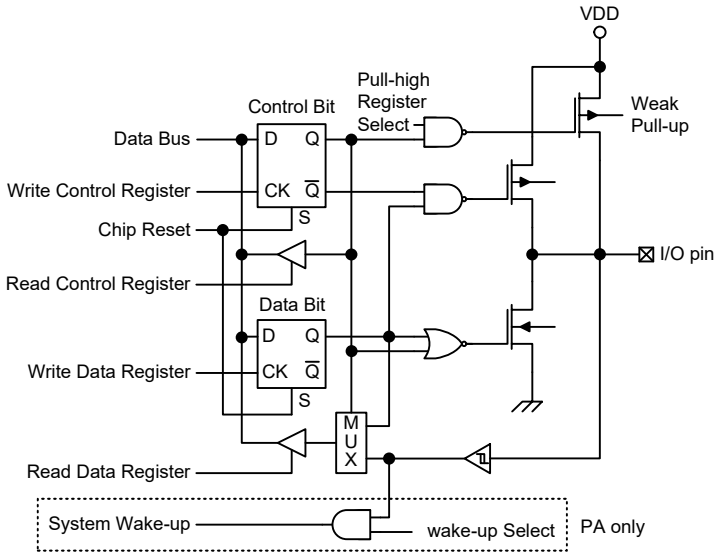
Bit 5~4 **PBS15~PBS14**: PB6 pin-shared control bit
00: PB6
01: VDDIO
10: PB6
11: PB6

Bit 3~2 **PBS13~PBS12**: PB5 pin-shared control bit
00: PB5
01: $\overline{\text{SPISCS}}$
10: PB5
11: PB5

Bit 1~0 **PBS11~PBS10**: PB4 pin-shared control bit
00: PB4
01: SPISCK
10: PB4
11: PB4

I/O Pin Structures

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



Logic Function Input/Output Structure

Programming Considerations

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

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

Timer Modules – TM

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

The general features of the Compact and Standard type TMs are described here with more detailed information provided in the individual Compact and Standard type TM sections.

Introduction

The devices contain from one to two TMs depending upon which device is selected and each individual TM can be categorized as a certain type, namely Compact Type TM or Standard Type TM. The common features to all of the Compact and Standard TMs will be described in this section and the detailed operation regarding each of the TM types will be described in separate sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

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

TM Function Summary

Device	CTM	STM
BH66F71652	10-bit CTM	—
BH66F71662	10-bit CTM	10-bit STM

TM Name/Type Reference

TM Operation

The different typs of TM offer a diverse range of functions, from simple timing operations to PWM signal generation. The key to understanding how the TM operates is to see it in terms of a free running count-up counter whose value is then compared with the value of pre-programmed internal comparators. When the free running count-up 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 the TM can originate from various sources. The selection of the required clock source is implemented using thexTCK2~xTCK0 bits in the xTM control register, where "x" stands for C or S type TM. The clock source can be a ratio of the system clock f_{SYS} or the internal high clock f_H, the f_{SUB} clock source or the external xTCK pin. The xTCK pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

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

TM External Pins

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

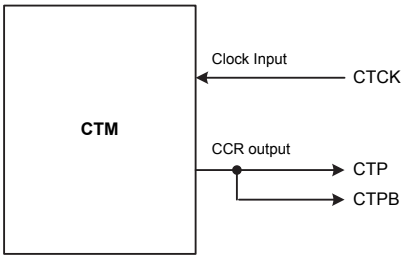
The other STM input pin, STPI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the STIO1~STIO0 bits in the STMC1 register.

The TMs each have two output pins with the label xTP and xTPB. The xTPB pin outputs the inverted signal of the xTP. When the xTM is in the Compare Match Output Mode, these pins can be controlled by the xTM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTP and xTPB output pins are also the pins where the TM generates the PWM output waveform.

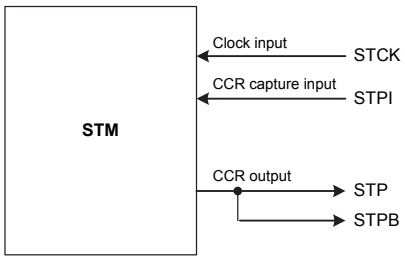
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using the relevant pin-shared function selection bits described in the Pin-shared Function section. The details of the pin-shared function selection are described in the pin-shared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

Device	CTM		STM	
	Input	Output	Input	Output
BH66F71652	CTCK	CTP, CTPB	—	—
BH66F71662	CTCK	CTP, CTPB	STCK, STPI	STP, STPB

TM External Pins



CTM Function Pin Block Diagram

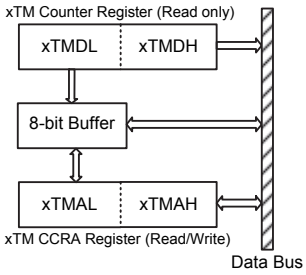


STM Function Pin Block Diagram

Programming Considerations

The TM Counter Registers and the Capture/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 xTMAL, 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:

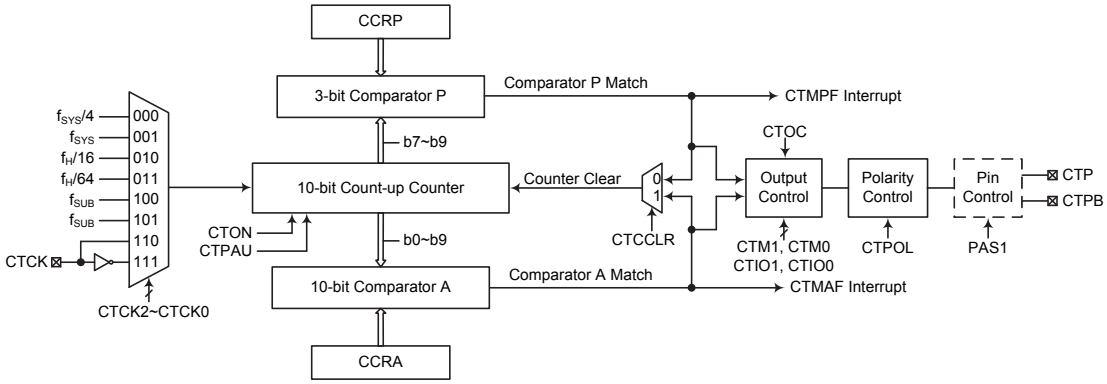
- Writing Data to CCRA
  - Step 1. Write data to Low Byte xTMAL
    - Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte xTMAH
    - Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte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

- Reading Data from the Counter Registers and CCRA
  - Step 1. Read data from the High Byte xTMDH, xTMAH
    – Here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - Step 2. Read data from the Low Byte xTMDL, xTMAL
    – This step reads data from the 8-bit buffer.

Compact Type TM – CTM

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

Device	CTM Core	CTM Input Pin	CTM Output Pin
BH66F71652/BH66F71662	10-bit CTM	CTCK	CTP, CTPB



Compact Type TM Block Diagram

Compact TM Operation

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

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

Compact Type TM Register Description

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

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

10-bit Compact TM Register List

- **CTMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CTPAU**: CTM Counter pause control
0: Run
1: Pause

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

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

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

Bit 3 **CTON**: CTM Counter on/off control
0: Off
1: On

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

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

Bit 2~0 **CTRP2~CTRP0**: CTM CCRP 3-bit register, compared with the CTM Counter bit 9~bit 7

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

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

• CTMC1 Register

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

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

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

Bit 5~4 **CTIO1~CTIO0**: Select CTM external pin (CTP) function

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

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

Timer/counter Mode:
Unused

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

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

using the CTOC bit otherwise no change will occur on the CTM output pin when a compare match occurs. After the CTM output pin changes state it can be reset to its initial level by changing the level of the CTON bit from low to high.

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

Bit 3 **CTOC**: CTM CTP output control bit

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

PWM Output Mode:
0: Active low
1: Active high

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

Bit 2 **CTPOL**: CTM CTP output polarity Control
0: Non-invert
1: Invert

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

Bit 1 **CTDPX**: CTM PWM period/duty control
0: CCRP - period; CCRA – duty
1: CCRP - duty; CCRA – period

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

Bit 0 **CTCCLR**: CTM Counter clear condition selection
0: CTM Comparatror P match
1: CTM Comparatror A match

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

- **CTMDL Register**

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

Bit 7~0 **D7~D0**: CTM Counter low byte register bit 7 ~ bit 0
CTM 10-bit Counter bit 7 ~ bit 0

• **CTMDH Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **D9~D8**: CTM Counter high byte register bit 1 ~ bit 0
CTM 10-bit Counter bit 9 ~ bit 8

• **CTMAL Register**

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

Bit 7~0 **D7~D0**: CTM CCRA low byte register bit 7 ~ bit 0
CTM 10-bit CCRA bit 7 ~ bit 0

• **CTMAH Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **D9~D8**: CTM CCRA high byte register bit 1 ~ bit 0
CTM 10-bit CCRA bit 9 ~ bit 8

Compact Type TM Operating Modes

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

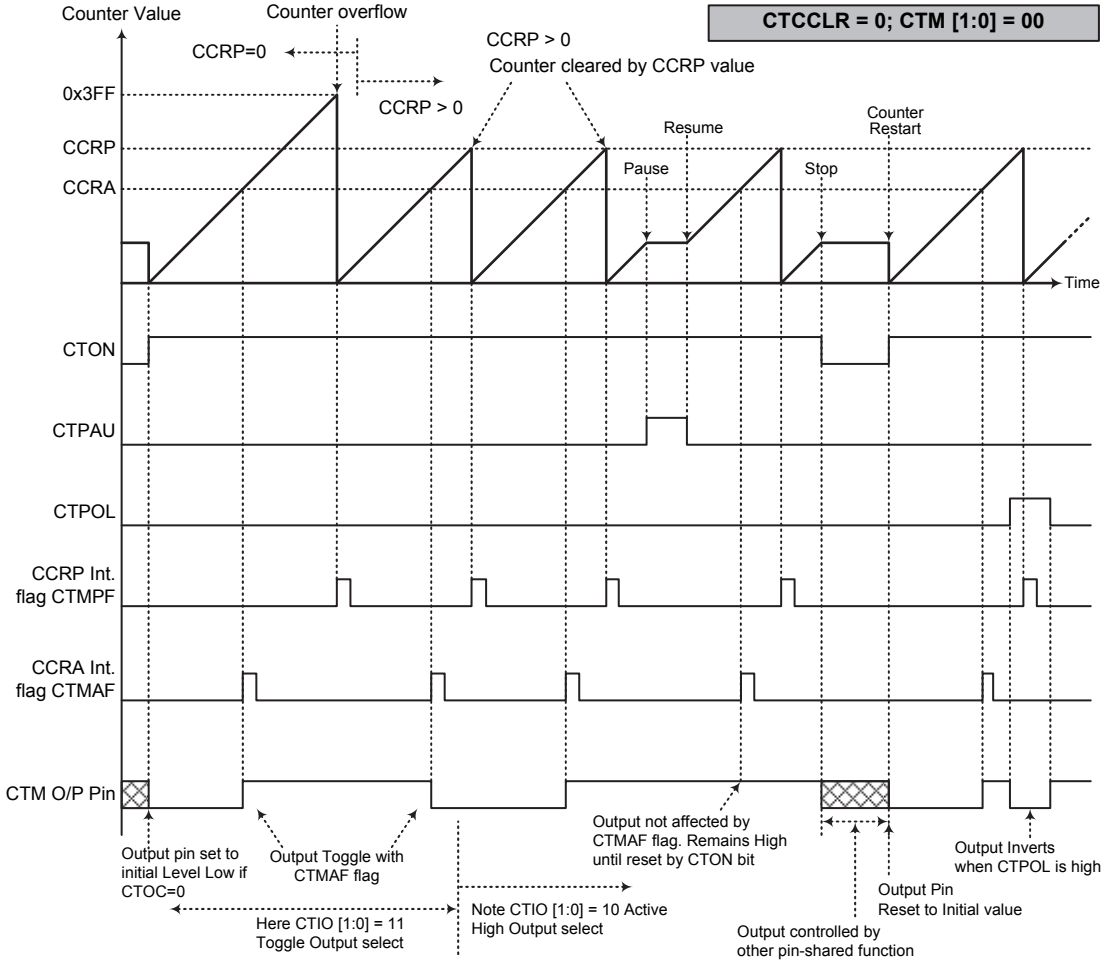
Compare Match Output Mode

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

If the CTCCLR bit in the CTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTCCLR is high no CTMPF interrupt request flag will be generated.

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

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

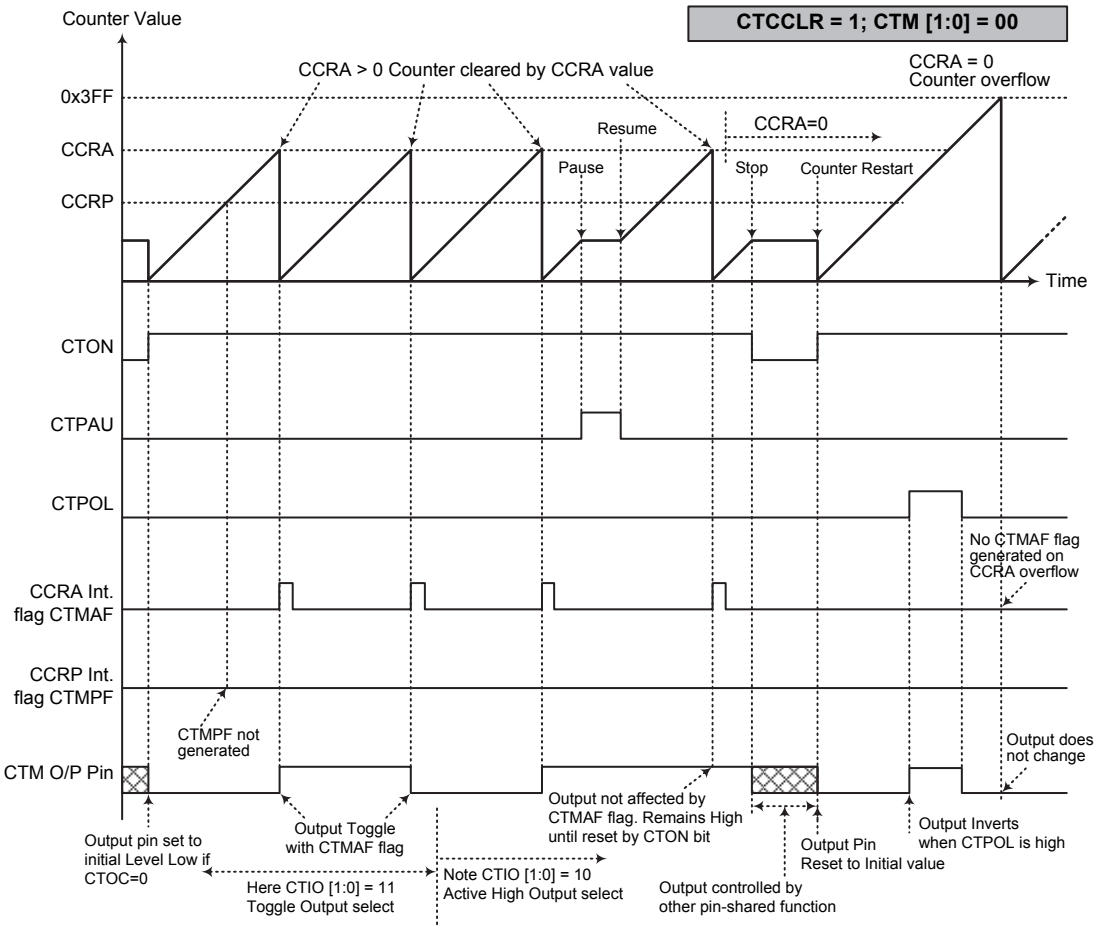


Compare Match Output Mode – CTCCLR=0

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

2. The CTM output pin controlled only by the CTMAF flag

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



Compare Match Output Mode – CTCCLR=1

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

2. The CTM output pin controlled only by the CTMAF flag
3. The output pin reset to initial state by a CTON rising edge
4. The CTMPF flags is not generated when CTCCLR=1

Timer/Counter Mode

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

PWM Output Mode

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

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the CTCCLR bit has no effect 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 CTDPX bit in the CTMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

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

• 10-bit CTM, PWM Output Mode, Edge-aligned Mode, CTDPX=0

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

If f_{SYS}=8MHz, CTM clock source is $f_{SYS}/4$, CCRP=2 and CCRA=128,

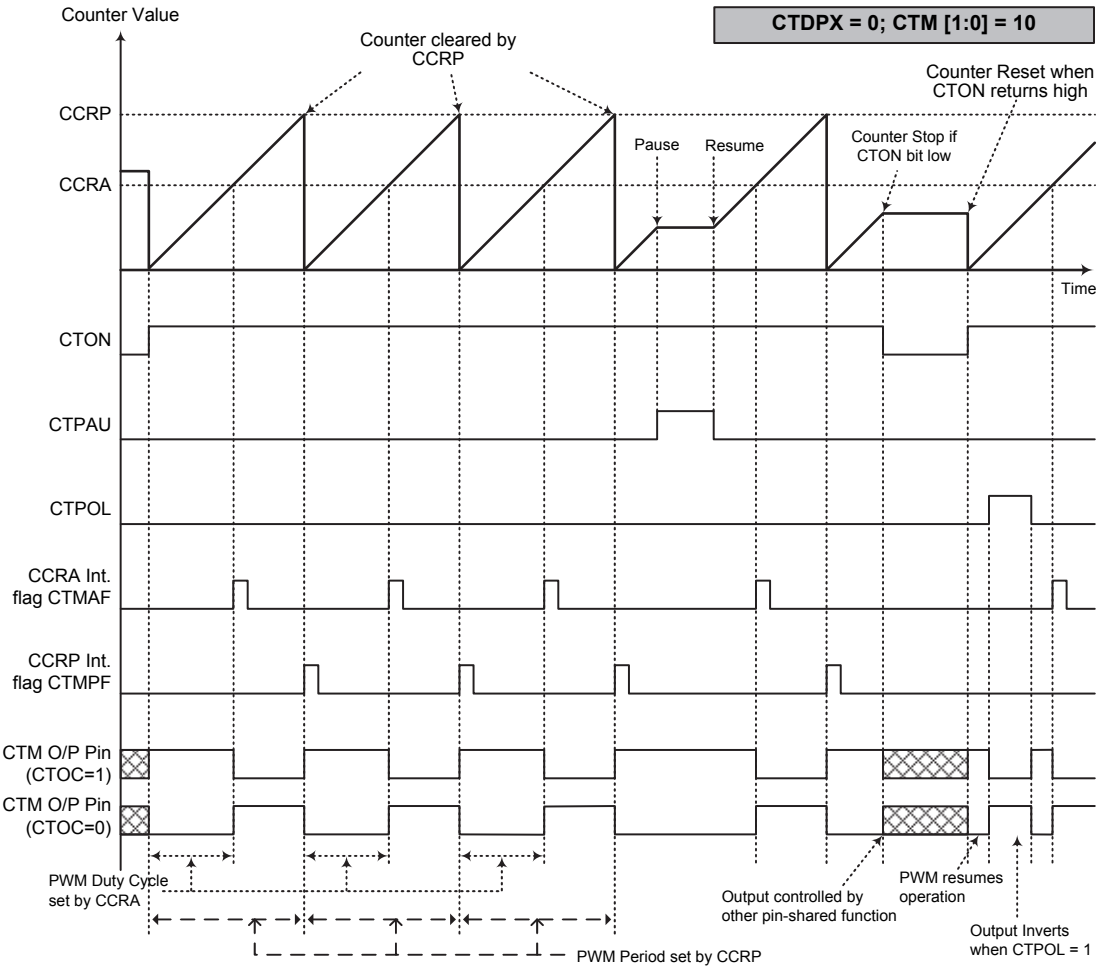
The CTM PWM output frequency=$(f_{SYS}/4)/(2\times128)=f_{SYS}/1024$=7.8125kHz, duty=128/(2×128)=50%.

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 CTM, PWM Output Mode, Edge-aligned Mode, CTDPX=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 CTM clock while the PWM duty cycle is defined by the CCRP register value.



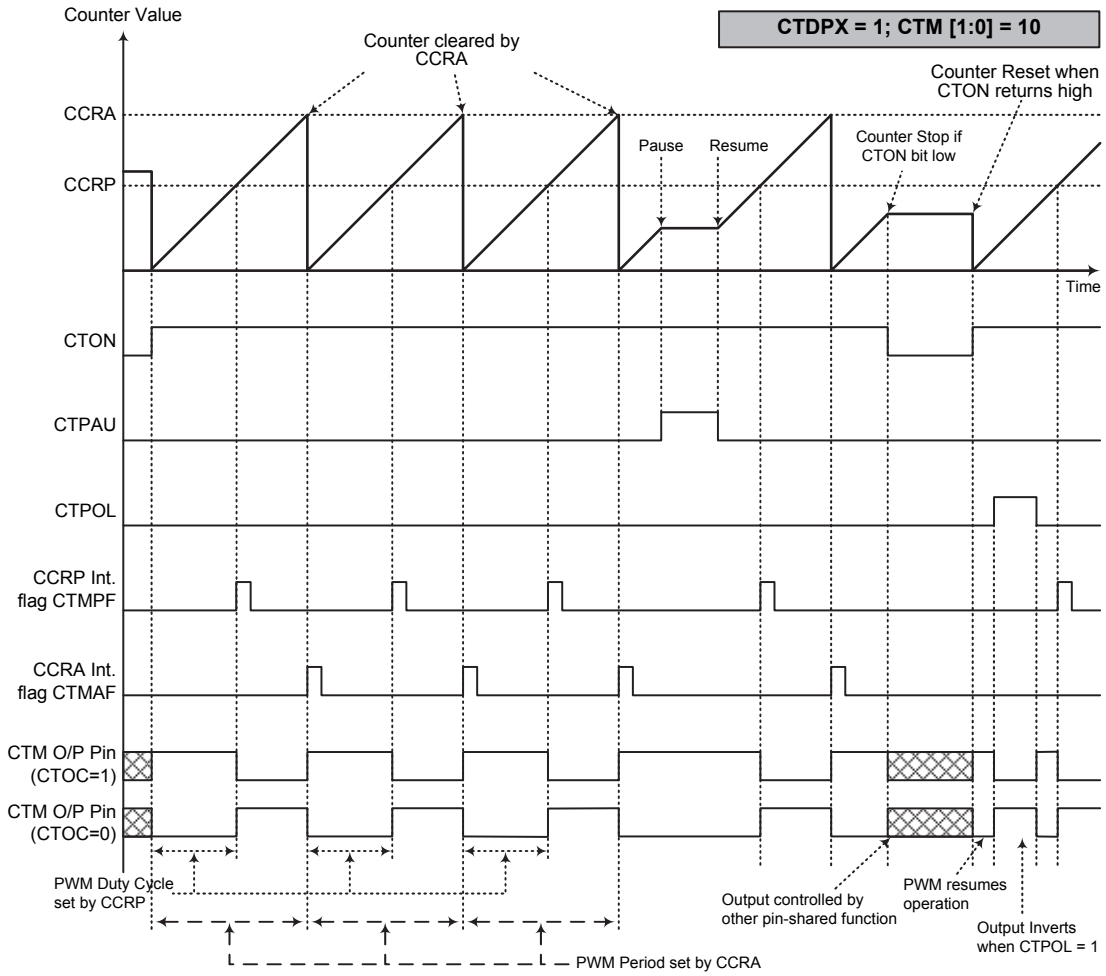
PWM Output Mode – CTDPX=0

Note: 1. Here CTDPX=0 – Counter cleared by CCRP

2. A counter clear sets PWM Period

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

4. The CTCCLR bit has no influence on PWM operation



PWM Output Mode – CTDPX=1

Note: 1. Here CTDPX=1 – Counter cleared by CCRA

2. A counter clear sets PWM Period

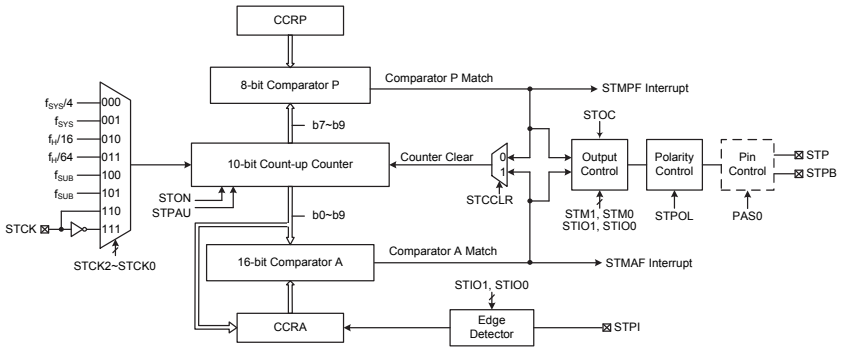
3. The internal PWM function continues even when CTIO[1:0]=00 or 01

4. The CTCCLR bit has no influence on PWM operation

Standard Type TM – STM

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

Device	STM Core	STM Input Pin	STM Output Pin
BH66F71662	10-bit STM	STCK, STPI	STP, STPB



10-bit Standard Type TM Block Diagram

Standard TM Operation

The size of Standard 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 the with highest 3 bits in the counter while the CCRA is the ten 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 STON 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 STM interrupt signal will also usually be generated. The Standard 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 the output pins. All operating setup conditions are selected using relevant internal registers.

Standard Type TM Register Description

Overall operation of the Standard 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 set the different operating and control modes as well as the three CCRP bits.

Register Name	Bit							
	7	6	5	4	3	2	1	0
STMC0	STPAU	STCK2	STCK1	STCK0	STON	STPR2	STPR1	STPR0
STMC1	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
STMDL	D7	D6	D5	D4	D3	D2	D1	D0
STMDH	—	—	—	—	—	—	D9	D8
STMAL	D7	D6	D5	D4	D3	D2	D1	D0
STMAH	—	—	—	—	—	—	D9	D8

10-bit Standard TM Register List

• **STMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	STPAU	STCK2	STCK1	STCK0	STON	STPR2	STPR1	STPR0
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 **STPAU**: STM 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 STM 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 **STCK2~STCK0**: STM 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: STCK rising edge clock
111: STCK falling edge clock

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

Bit 3 **STON**: STM counter on/off control
0: Off
1: On

This bit controls the overall on/off function of the STM. Setting the bit high enables the counter to run while clearing the bit disables the STM. Clearing this bit to zero will stop the counter from counting and turn off the STM 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 STM is in the Compare Match Output Mode, PWM Output Mode or Single Pulse Output Mode then the STM output pin will be reset to its initial condition, as specified by the STOC bit, when the STON bit changes from low to high.

Bit 2~0 **STRP2~STRP0**: STM CCRP 3-bit register, compared with the STM Counter bit 9 ~ bit 7
Comparator P Match Period:
000: 1024 STM clocks
001: 128 STM clocks
010: 256 STM clocks
011: 384 STM clocks
100: 512 STM clocks
101: 640 STM clocks
110: 768 STM clocks
111: 896 STM 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 STCCLR bit is set to zero. Setting the STCCLR 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.

• **STMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
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 **STM1~STM0**: STM operating mode selection
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits are used to set the required operating mode for the STM. To ensure reliable operation the STM should be switched off before any changes are made to the STM1 and STM0 bits. In the Timer/Counter Mode, the STM output pin state is undefined.

Bit 5~4 **STIO1~STIO0**: STM external pin (STP or STPI) function selection

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

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

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

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a compare match occurs from the Comparator A. The STM output pin can be set 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 STM output pin should be set using the STOC bit in the STMC1 register. Note that the output level requested by the STIO1 and STIO0 bits must be different from the initial value setup using the STOC bit otherwise no change will occur on the STM output pin when a compare match occurs. After the STM output pin changes state, it can be reset to its initial level by changing the level of the STON bit from low to high.

In the PWM Output Mode, the STIO1 and STIO0 bits determine how the STM 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 STIO1 and STIO0 bits only after the STM has been switched off. Unpredictable PWM outputs will occur if the STIO1 and STIO0 bits are changed when the STM is running.

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

Bit 2 **STPOL**: STM STP output polarity control
0: Non-invert
1: Invert

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

Bit 1 **STDPX**: STM 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 **STCCLR**: STM 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 Standard TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the STCCLR 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 STCCLR bit is not used in the PWM Output Mode, Single Pulse Output Mode or Capture Input Mode.

• STMDL 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**: STM Counter Low Byte Register bit 7 ~ bit 0
STM 10-bit Counter bit 7 ~ bit 0

• STMDH 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**: STM Counter High Byte Register bit 1 ~ bit 0
STM 10-bit Counter bit 9 ~ bit 8

• **STMAL 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**: STM CCRA Low Byte Register bit 7 ~ bit 0
STM 10-bit CCRA bit 7 ~ bit 0

• **STMAH 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 Umplemented, read as “0”

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

Standard Type TM Operation Modes

The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

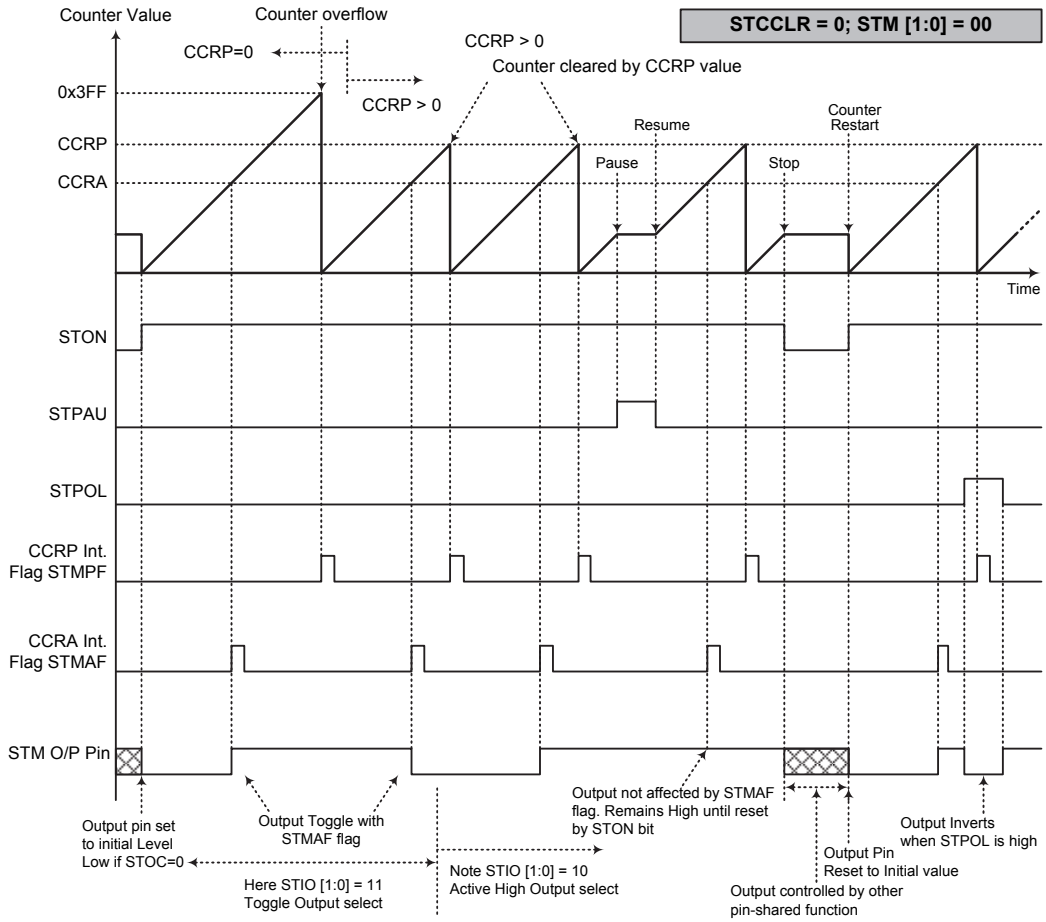
Compare Match Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 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 STCCLR 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 STMAF and STMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the STCCLR bit in the STMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when STCCLR is high no STMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”.

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

As the name of the mode suggests, after a comparison is made, the STM output pin, will change state. The STM output pin condition however only changes state when a STMAF interrupt request flag is generated after a compare match occurs from Comparator A. The STMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the STM output pin. The way in which the STM output pin changes state are determined by the condition of the STIO1 and STIO0 bits in the STMC1 register. The STM output pin can be selected using the STIO1 and STIO0 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 STM output pin, which is configured after the STON bit changes from low to high, is configured using the STOC bit. Note that if the STIO1 and STIO0 bits are zero then no pin change will take place.

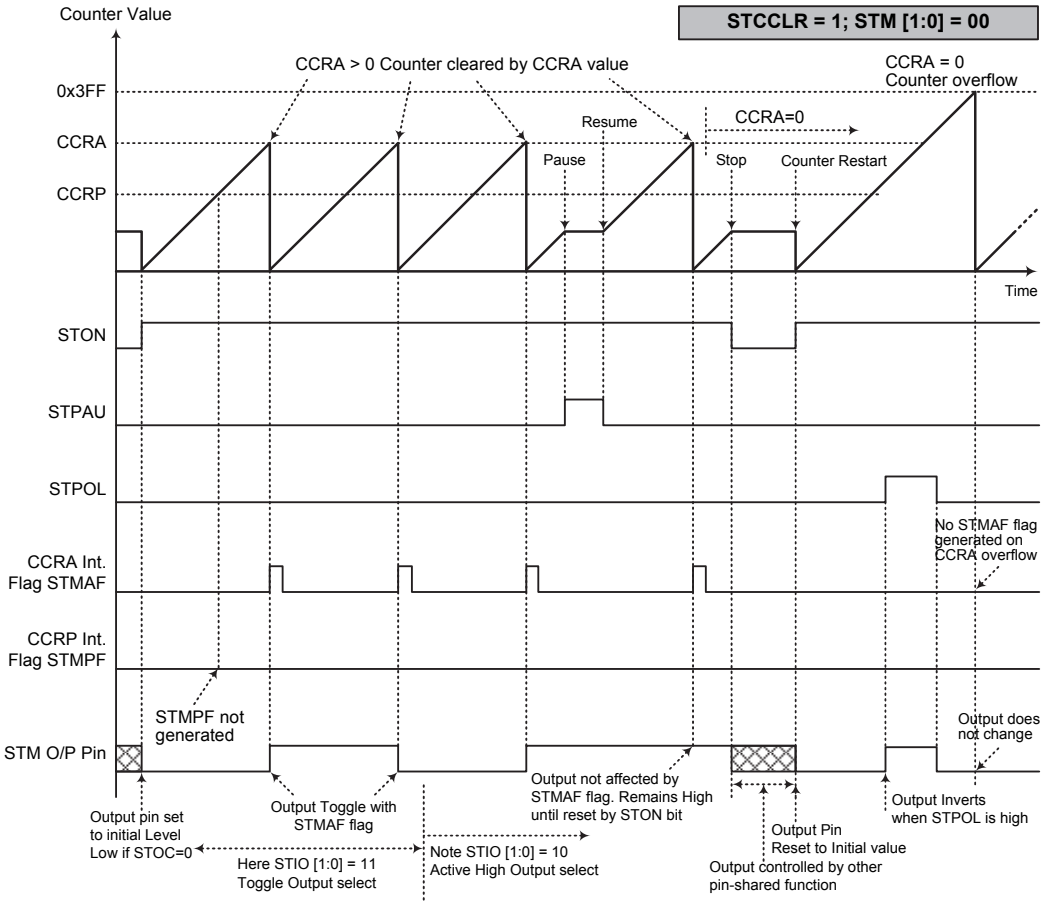


Compare Match Output Mode – STCCLR=0

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

2. The STM output pin is controlled only by the STMAF flag

3. The output pin is reset to itsinitial state by a STON bit rising edge



Compare Match Output Mode – STCCLR=1

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

2. The STM output pin is controlled only by the STMAF flag
3. The output pin is reset to its initial state by a STON bit rising edge
4. The STMPF flag is not generated when STCCLR=1

Timer/Counter Mode

To select this mode, bits STM1 and STM0 in the STMC1 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 STM 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 STM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared functions.

PWM Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to "10" respectively and also the STIO1 and STIO0 bits should be set to 10 respectively. The PWM function within the STM 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 STM 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 STCCLR 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 STDPX bit in the STMC1 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 STOC bit in the STMC1 register is used to select the required polarity of the PWM waveform while the two STIO1 and STIO0 bits are used to enable the PWM output or to force the STM output pin to a fixed high or low level. The STPOL bit is used to reverse the polarity of the PWM output waveform.

- **10-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=0**

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

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

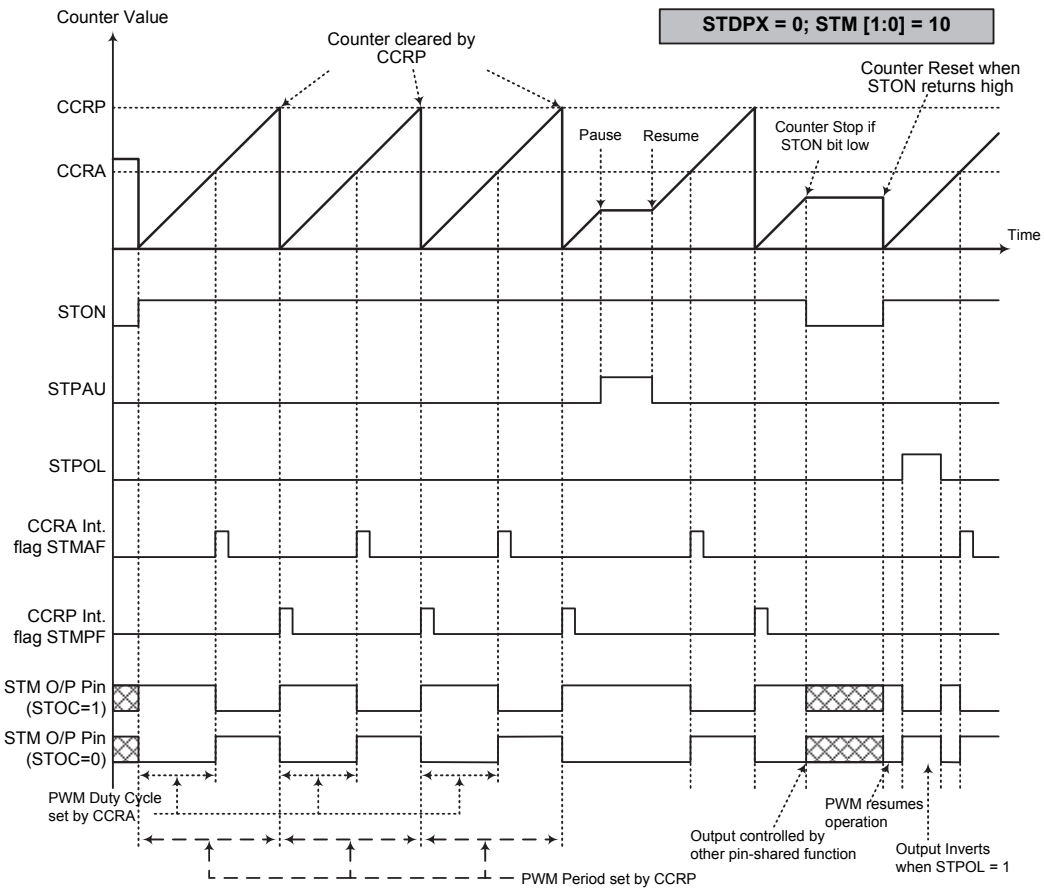
The STM PWM output frequency=(f_{SYS}/4)/512=f_{SYS}/2048=3.9063kHz, duty=128/512=25%.

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

- **10-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=1**

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

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



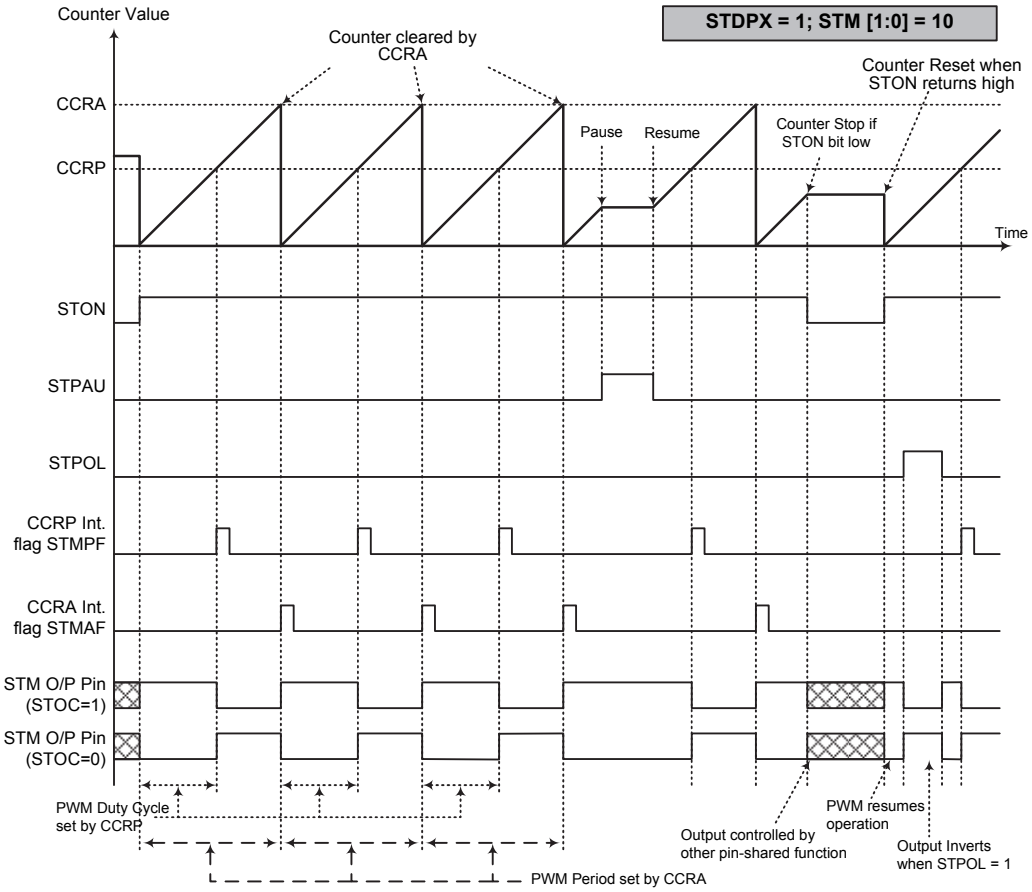
PWM Output Mode – STDPX=0

Note: 1. Here STDPX=0 – Counter cleared by CCRP

2. A counter clear sets the PWM Period

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

4. The STCCLR bit has no influence on PWM operation



PWM Output Mode – STDPX=1

Note: 1. Here STDPX=1 – Counter cleared by CCRA

2. A counter clear sets the PWM Period

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

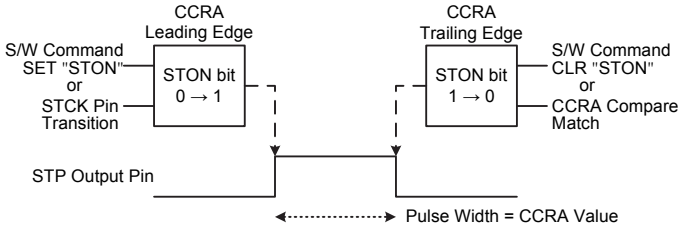
4. The STCCLR bit has no influence on PWM operation

Single Pulse Output Mode

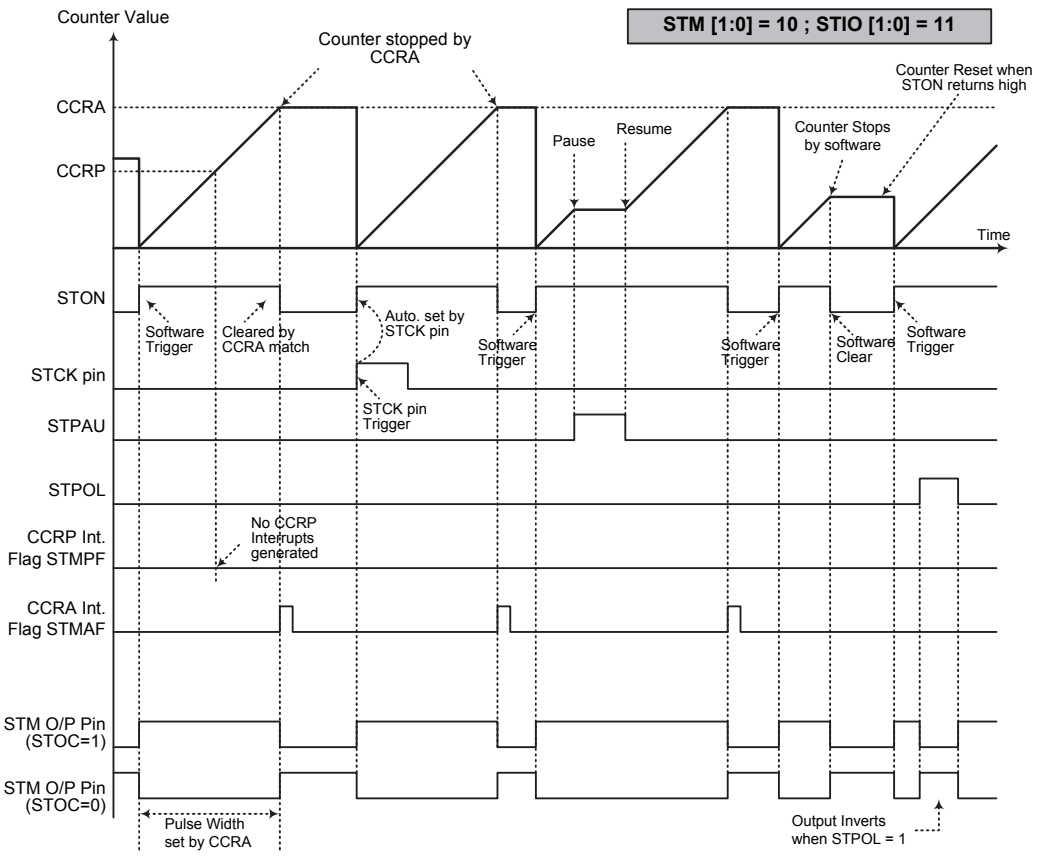
To select this mode, bits STM1 and STM0 in the STMC1 register should be set to "10" respectively and also the STIO1 and STIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

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

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



Single Pulse Generation



Single Pulse Output Mode

Note: 1. Counter stopped by CCRA

2. CCRP is not used

3. The pulse triggered by the STCK pin or by setting the STON bit high

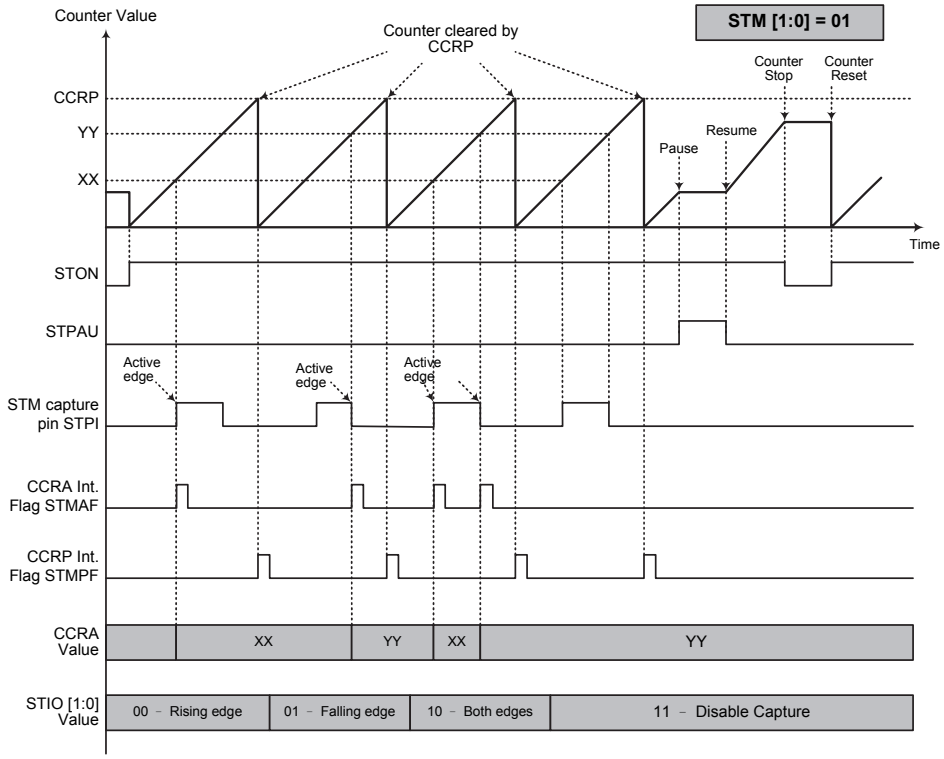
4. A STCK pin active edge will automatically set the STON bit high

5. In the Single Pulse Output Mode, STIO[1:0] must be set to “11” and can not be changed

Capture Input Mode

To select this mode bits STM1 and STM0 in the STMC1 register should be set to "01" respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the STPI pin, whose active edge can be a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the STIO1 and STIO0 bits in the STMC1 register. The counter is started when the STON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the STPI pin the present value in the counter will be latched into the CCRA registers and a STM interrupt generated. Irrespective of what events occur on the STPI pin the counter will continue to free run until the STON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a STM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The STIO1 and STIO0 bits can select the active trigger edge on the STPI pin to be a rising edge, falling edge or both edge types. If the STIO1 and STIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the STPI pin, however it must be noted that the counter will continue to run. The STCCLR and STDPX bits are not used in this Mode.



Capture Input Mode

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

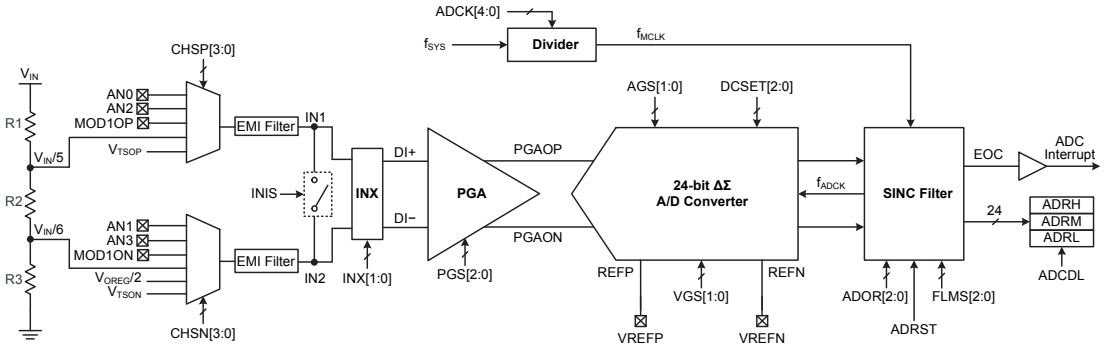
Analog to Digital Converter

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

A/D Converter Overview

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

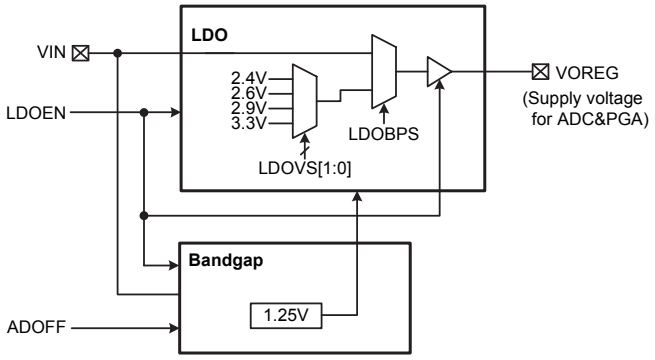
In addition, PGA gain control, A/D converter gain control and A/D converter reference gain control determine the amplification gain for A/D converter input signal. The designer can select the best gain combination for the desired amplification applied to the input signal. The following block diagram illustrates the A/D converter basic operational function. The A/D converter input channel can be arranged as 4 single-ended A/D converter input channels or 2 differential input channels. The input signal can be amplified by PGA before entering the 24-bit Delta Sigma A/D converter. The A/D converter module will output one bit converted data to the SINC filter which can transform the converted one-bit data to 24 bits and store them into the specific data registers. Additionally, the devices also provide a temperature sensor to compensate the A/D converter deviation caused by the temperature. With high accuracy and performance, the devices are very suitable for differential output sensor applications such as weight measurement scales and other related products.



A/D Converter Structure

Internal Power Supply

The devices contain an LDO for the regulated power supply. The accompanying block diagram illustrates the basic functional operation. The internal LDO can provide a fixed voltage for the PGA, A/D converter or external components. There are four LDO voltage levels, 2.4V, 2.6V, 2.9V or 3.3V, determined by the LDOVS1~LDOVS0 bits in the PWRC register. The LDO function can be controlled by the LDOEN and can be powered off to reduce overall power consumption.



Internal Power Supply Block Diagram

Control Bits		Output Voltage	
ADOFF	**LDOEN**	**Bandgap**	**V_{OREG}**
1	0	Off	Disable
1	1	On	Enable
0	0	On	Disable
0	1	On	Enable

Power Control Table

- **PWRC Register**

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

Bit 7 **LDOEN**: LDO function control
0: Disable
1: Enable
If the LDO is disabled, there will be no power consumption and the LDO output pin will remain at a low level using a weak internal pull-low resistor.

Bit 6~3 Unimplemented, read as "0"

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

Bit 1~0 **LDOVS1~LDOVS0**: LDO output voltage selection
00: 2.4V
01: 2.6V
10: 2.9V
11: 3.3V

A/D Converter Data Rate Definition

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

$$\text{Data Rate} = \frac{f_{ADCK}}{CHOP \times OSR} = \frac{f_{MCLK}/N}{CHOP \times OSR} = \frac{f_{MCLK}}{N \times CHOP \times OSR}$$

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

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

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

CHOP: Sampling data amount doubling function control equal to 1 or 2 determined by the FLMS bit field.

OSR: Oversampling rate determined by the ADOR bit field.

For example, if a data rate of 8Hz is desired, an f_{MCLK} clock source with a frequency of 4MHz can be selected. Then set the FLMS field to "000" to obtain an "N" equal to 30 and "CHOP" equal to 2. Finally, set the ADOR field to "001" to select an oversampling rate equal to 8192. Therefore, the Data Rate=4MHz/(30×2×8192)=8Hz.

Note that the A/D converter has a notch rejection function for AC power supplies with a frequency of 50Hz or 60Hz when the data rate is equal to 10Hz.

A/D Converter Register Description

Overall operation of the A/D converter is controlled by using a series of registers. Three read only registers exist to store the A/D converter data 24-bit value. A control register named as PWRC is used to control the required bias and supply voltages for PGA and A/D converter and is described in the "Internal Power Supply" section. The remaining registers are control registers which set up the gain selections and control functions of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PWRC	LDOEN	—	—	—	—	LDOBPS	LDOVS1	LDOVS0
PGAC0	—	VGS1	VGS0	AGS1	AGS0	PGS2	PGS1	PGS0
PGAC1	—	INIS	INX1	INX0	DCSET2	DCSET1	DCSET0	—
PGACS	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
ADCS	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
ADCR0	ADRST	ADSLP	ADOFF	ADOR2	ADOR1	ADOR0	—	—
ADCR1	FLMS2	FLMS1	FLMS0	—	—	ADCDL	EOC	—
ADRL	D7	D6	D5	D4	D3	D2	D1	D0
ADRM	D15	D14	D13	D12	D11	D10	D9	D8
ADRH	D23	D22	D21	D20	D19	D18	D17	D16

A/D Converter Register List

Programmable Gain Amplifier Registers – PGAC0, PGAC1, PGACS

There are three registers related to the programmable gain control, PGAC0, PGAC1 and PGACS. The PGAC0 resister is used to select the PGA gain, A/D Converter gain and the A/D Converter reference gain. The PGAC1 register is used to define the input connection and differential input offset voltage adjustment control. In addition, the PGACS register is used to select the PGA inputs. Therefore, the CHSP3~CHSP0 and CHSN3~CHSN0 bits determine which analog channel input pins, temperature detector inputs or internal power supply are actually connected to the internal differential A/D converter.

- **PGAC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	VGS1	VGS0	AGS1	AGS0	PGS2	PGS1	PGS0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~5 **VGS1~VGS0**: REFP/REFN differential reference voltage gain selection
00: VREFGN=1
01: VREFGN=1/2
10: VREFGN=1/4
11: Reserved

Bit 4~3 **AGS1~AGS0**: A/D converter PGAOP/PGAON differential input signal gain selection
00: ADGN=1
01: ADGN=2
10: ADGN=4
11: Reserved

Bit 2~0 **PGS2~PGS0**: PGA DI+/DI- differential channel input gain selection
000: PGAGN=1
001: PGAGN=2
010: PGAGN=4
011: PGAGN=8
100: PGAGN=16
101: PGAGN=32
110: PGAGN=64
111: PGAGN=128

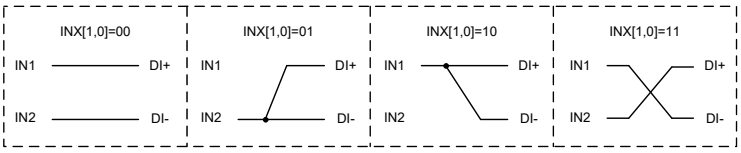
• PGAC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	INIS	INX1	INX0	DCSET2	DCSET1	DCSET0	—
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	—	0	0	0	0	0	0	—

Bit 7 Unimplemented, read as "0"

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

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



Bit 3~1 **DCSET2~DCSET0**: Differential input signal PGAOP/PGAON offset selection
000: DCSET = +0V
001: DCSET = +0.25 × ΔVR_I
010: DCSET = +0.5 × ΔVR_I
011: DCSET = +0.75 × ΔVR_I
100: DCSET = +0V
101: DCSET = -0.25 × ΔVR_I
110: DCSET = -0.5 × ΔVR_I
111: DCSET = -0.75 × ΔVR_I
The voltage, ΔVR_I, is the differential reference voltage which is amplified by specific gain selection based on the selected inputs.

Bit 0 Unimplemented, read as "0"

• PGACS Register

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

Bit 7~4 **CHSN3~CHSN0**: Negative input end IN2 selection
0000: AN1
0001: AN3
0010: MOD1ON
0011: $V_{OREG}/2$

0100~0101: Reserved
0110: $V_{IN}/6$
0111: Temperature sensor output – V_{TSON}
1xxx: Reserved

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

Bit 3~0 **CHSP3~CHSP0**: Positive input end IN1 selection
0000: AN0
0001: AN2
0010: MOD1OP
0011~0101: Reserved
0110: $V_{IN}/5$
0111: Temperature sensor output – V_{TSOP}
1xxx: Reserved

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

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

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

- **ADRL Register**

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

"x": unknown

Bit 7~0 A/D conversion data register bit 7 ~ bit 0

- **ADRM Register**

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

"x": unknown

Bit 7~0 A/D conversion data register bit 15 ~ bit 8

- **ADRH Register**

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

"x": unknown

Bit 7~0 A/D conversion data register bit 23 ~ bit 16

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

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

- **ADCR0 Register**

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

Bit 7 **ADRST**: A/D converter software reset control
0: Disable
1: Enable

This bit is used to reset the A/D converter internal digital SINC filter. This bit is cleared to zero for normal A/D converter operation. However, if set high, the internal digital SINC filter will be reset and the current A/D converted data will be aborted. A new A/D data conversion process will not be initiated until this bit is cleared to zero again.

Bit 6 **ADSLP**: A/D converter sleep mode control bit
0: Normal mode
1: Sleep mode

This bit is used to determine whether the A/D converter enters the sleep mode or not when the A/D converter is powered on by clearing the ADOFF to zero. When the A/D converter is powered on and the ADSLP bit is low, the A/D converter will operate normally. However, the A/D converter will enter the sleep mode if the ADSLP bit is set high as the A/D converter has been powered on. The whole A/D converter circuit will be switched off except the PGA and internal Bandgap circuit to reduce overall power consumption.

Bit 5 **ADOFF**: A/D converter module power on/off control bit
0: Power on
1: Power off

This bit controls the A/D converter power on/off function. This bit should be cleared to zero to enable the A/D converter. If the bit is set high then the A/D converter will be switched off reducing the device power consumption. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.

It is recommended to set the ADOFF bit high before the device enters the IDLE/SLEEP mode to save power. Setting the ADOFF bit high will power down the A/D converter module regardless of the ADSLP and ADRST bit settings.

Bit 4~2 **ADOR2~ADOR0**: A/D conversion oversampling rate selection
000: Oversampling rate OSR=16384
001: Oversampling rate OSR=8192
010: Oversampling rate OSR=4096
011: Oversampling rate OSR=2048
100: Oversampling rate OSR=1024
101: Oversampling rate OSR=512
110: Oversampling rate OSR=256
111: Oversampling rate OSR=128

Bit 1~0 Unimplemented, read as “0”

• **ADCR1 Register**

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

Bit 7~5 **FLMS2~FLMS0**: A/D converter clock frequency and sampled data doubling function control
000: CHOP=2, $f_{ADCK}=f_{MCLK}/30$
010: CHOP=2, $f_{ADCK}=f_{MCLK}/12$
100: CHOP=1, $f_{ADCK}=f_{MCLK}/30$
110: CHOP=1, $f_{ADCK}=f_{MCLK}/12$
Other values: Reserved
When the CHOP bit is equal to 2, it means that the sampled data will be doubled for the normal conversion mode. However, it can be regarded as a low latency conversion mode if the CHOP bit is equal to 1, which means that the sampled data doubling function is disabled.

Bit 4~3 Unimplemented, read as "0"

Bit 2 **ADCDL**: A/D converted data latch function enable control
0: Disable data latch function
1: Enable data latch function
If the A/D converted data latch function is enabled, the latest converted data value will be latched and will not be updated by any subsequent conversion results until this function is disabled. Although the converted data is latched into the data registers, the A/D converter circuits remain operational, but will not generate an interrupt and the EOC will not change. It is recommended that this bit should be set high before reading the converted data from the ADRL, ADRM and ADRH registers. After the converted data has been read out, the bit can then be cleared to zero to disable the A/D converter data latch function and allow further conversion values to be stored. In this way, the possibility of obtaining undesired data during A/D converter conversions can be prevented.

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

Bit 0 Unimplemented, read as "0"

• **ADCS Register**

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

Bit 7~5 Unimplemented, read as "0"

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

A/D Converter Operation

The A/D Converter provides four operating modes, which are the Normal mode, Power down mode, Sleep mode and Reset mode, controlled respectively by the ADOFF, ADSLP and ADRST bits in the ADCR0 register. The following table illustrates the operating mode selection.

Control Bits				Operating mode	Description
LDOEN	ADOFF	ADSLP	ADRST		
0	1	x	x	Power down mode	Bandgap off, LDO off, PGA off, ADC off, Temperature sensor off, SINC filter off
1	1	x	x	Power down mode	Bandgap on, LDO on, PGA off, ADC off, Temperature sensor off, SINC filter off
0	0	1	x	Sleep mode (External voltage must be supplied on LDO output pin)	Bandgap on, LDO off, PGA on, ADC off, Temperature sensor off, SINC filter on
0	0	0	0	Normal mode (External voltage must be supplied on LDO output pin)	Bandgap on, LDO off, PGA on, ADC on, Temperature sensor on/off(1), SINC filter on
0	0	0	1	Reset mode (External voltage must be supplied on LDO output pin)	Bandgap on, LDO off, PGA on, ADC on, Temperature sensor on/off(1), SINC filter Reset
1	0	1	x	Sleep mode	Bandgap on, LDO on, PGA on, ADC off, Temperature sensor off, SINC filter on
1	0	0	0	Normal mode	Bandgap on, LDO on, PGA on, ADC on, Temperature sensor on/off(1), SINC filter on
1	0	0	1	Reset mode	Bandgap on, LDO on, PGA on, ADC on, Temperature sensor on/off(1), SINC filter Reset

Note: 1. The Temperature Sensor can be switched on or off by configuring the CHSN[3:0] or CHSP[3:0] bits.
2. "x" means unknown.

A/D Operating Mode Summary

To enable the A/D Converter, the first step is to disable the A/D converter power down and sleep mode by clearing the ADOFF and ADSLP bits to make sure the A/D Converter is powered on. The ADRST bit in the ADCR0 register is used to start and reset the A/D converter after power on. When the microcontroller changes this bit from low to high and then low again, an analog to digital conversion in the SINC filter will be initiated. After this setup is completed, the A/D Converter is ready for operation. These three bits are used to control the overall start operation of the internal analog to digital converter.

The EOC bit in the ADCR1 register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set high by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D converter interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D converter internal interrupt signal will direct the program flow to the associated A/D converter internal interrupt address for processing. If the A/D converter internal interrupt is disabled, the microcontroller can poll the EOC bit in the ADCR1 register to check whether it has been set to "1" as an alternative method of detecting the end of an A/D conversion cycle. The A/D converted data will be updated continuously by the new converted data. If the A/D converted data latch function is enabled, the latest converted data will be latched and the following new converted data will be discarded until this data latch function is disabled.

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

Summary of A/D Conversion Steps

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

- Step 1
  Enable the power LDO for PGA and ADC.
- Step 2
  Select the PGA, ADC, reference voltage gains by the PGAC0 register.
- Step 3
  Select the PGA settings for input connection and input offset by the PGAC1 register.
- Step 4
  Select the required A/D conversion clock source by correctly programming bits ADCK4~ADCK0 in the ADCS register.
- Step 5
  Select A/D conversion oversampling data rate and A/D clock by configuring the ADOR[2:0] bits in the ADCR0 register and FLMS[2:0] bits respectively in the ADCR1 register.
- Step 6
  Select which channel is to be connected to the internal PGA by correctly programming the CHSP3~CHSP0 and CHSN3~CHSN0 bits which are contained in the PGACS register.
- Step 7
  Release the power down mode and sleep mode by clearing the ADOFF and ADSLP bits in ADCR0 register.
- Step 8
  Reset the A/D converter by setting the ADRST bit in the ADCR0 register to high and then clear this bit to zero to release the reset status.
- Step 9
  If the A/D converter interrupt is to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bit, ADE, must both be set high to do this.
- Step 10
  To check when the analog to digital conversion process is complete, the EOC bit in the ADCR1 register can be polled. The conversion process is complete when this bit goes high. When this occurs the A/D converter data registers ADRL, ADRM and ADRH can be read to obtain the conversion value. As an alternative method, if the A/D converter and global interrupts are enabled and the stack is not full, the program can wait for an A/D converter interrupt to occur.

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

Programming Considerations

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

A/D Converter Transfer Function

The devices contain a 24-bit Delta Sigma A/D converter and its full-scale converted digitized value is from 8388607 to -8388608 in decimal value. The converted data format is formed using a two's complement binary value. The MSB of the converted data is the signed bit. Since the full-scale analog input value is equal to the amplified value of the differential reference input voltage, ΔVR_I, this gives a single bit analog input value of ΔVR_I divided by 8388608.

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

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

$$\Delta SI\_I = (PGAGN \times ADGN \times \Delta DI\pm) + DCSET$$

$$\Delta VR\_I = VREFGN \times \Delta VR\pm$$

$$ADC\_Conversion\_Data = (\Delta SI\_I / \Delta VR\_I) \times K$$

Where K is equal to 2^{23}

Note: 1. The PGAGN, ADGN, VREFGN values are decided by the PGS[2:0], AGS[1:0], VGS[1:0] control bits.
2. ΔSI_I: Differential Input Signal after amplification and offset adjustment.
3. PGAGN: Programmable Gain Amplifier gain
4. ADGN: A/D Converter gain
5. VREFGN: Reference voltage gain
6. ΔDI±: Differential input signal derived from external channels or internal signals
7. DCSET: Offset voltage
8. ΔVR±: Differential Reference voltage
9. ΔVR_I: Differential Reference input voltage after amplification

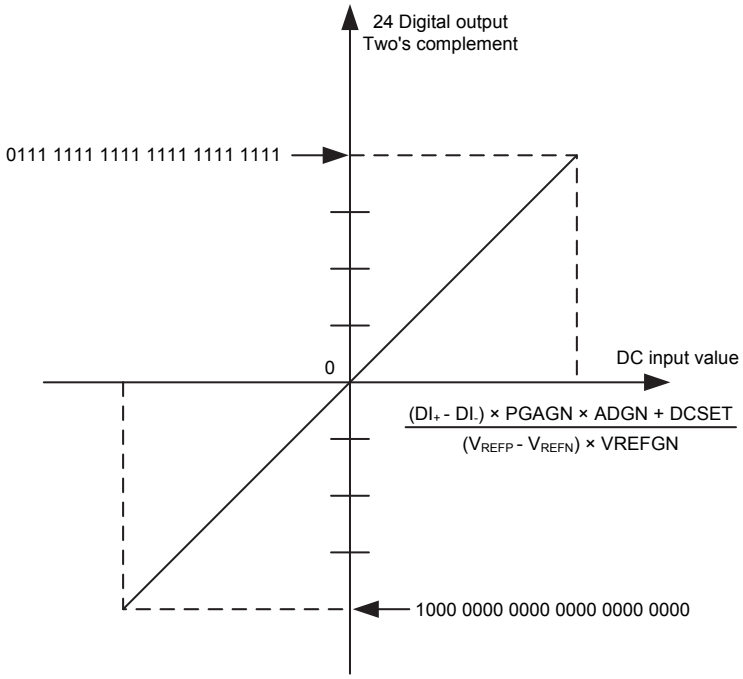
Due to the digital system design of the Delta Sigma A/D Converter, the maximum A/D converted value is 8388607 and the minimum value is -8388608. Therefore, there is a middle value of 0. The ADC_Conversion_Data equation illustrates this range of converted data variation.

A/D Conversion Data (2's complement, Hexadecimal)	Decimal Value
0x7FFFFF	8388607
0x800000	-8388608

A/D Conversion Data Range

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

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



A/D Converted Data

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

A/D Converted Data to Voltage

The converted data can be recovered using the following equations:

If MSB=0 - Positive Converted data

$$\text{Input Voltage} = \frac{(\text{Converted\_data}) \times \text{LSB} - \text{DCSET}}{\text{PGAGN} \times \text{ADGN}}$$

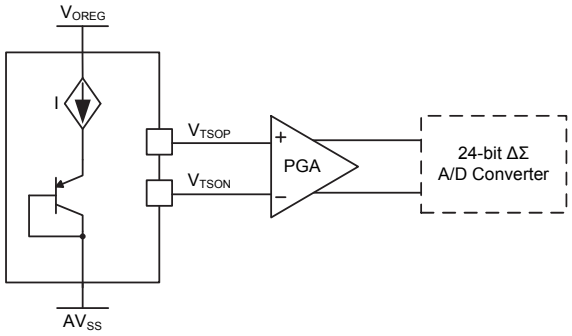
If the MSB=1 - Negative Converted data

$$\text{Input voltage} = \frac{(\text{Two's\_complement\_of\_Converted\_data}) \times \text{LSB} - \text{DCSET}}{\text{PGAGN} \times \text{ADGN}}$$

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

Temperature Sensor

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



Temperature Sensor Structure

A/D Conversion Programming Example

Example: Using an EOC polling method to detect the end of conversion

```
#include BH66F71652.inc
data .section   'data'
     adc_result_data_l db ?
     adc_result_data_m db ?
     adc_result_data_h db ?
code .section   'code'
start:
     clr ADE           ; disable ADC interrupt
     mov a, 83H        ; Power control for PGA, ADC
     mov PWRC, a       ; PWRC=10000011, LDO enable, LDO Bypass disable, LDO output voltage: 3.3V
     mov a, 00H
     mov PGAC0, a      ; PGA gain=1, ADC gain=1, VREF gain=1
     mov a, 00H
     mov PGAC1, a      ; INIS, INX, DCSET in default value
     clr ADOR2         ; for 10Hz output data rate, ADOR[2:0]=001, FLMS[2:0]=000
     clr ADOR1
     set ADOR0
     clr FLMS2
     clr FLMS1
     clr FLMS0
     clr ADOFF         ; ADC exit power down mode
     set ADRST         ; ADC in reset mode
     clr ADRST         ; ADC in conversion (continuous mode)
     clr EOC           ; Clear "EOC" flag
loop:
     snz EOC                    ; Polling "EOC" flag
     jmp loop                   ; Wait for read data
     clr adc_result_data_h
     clr adc_result_data_m
     clr adc_result_data_l
     set ADCDL                  ; enable data latch
     mov a, ADRL
     mov adc_result_data_l, a   ; Get Low byte ADC value
     mov a, ADRM
     mov adc_result_data_m, a   ; Get Middle byte ADC value
     mov a, ADRH
```

```
    mov adc_result_data_h, a    ; Get High byte ADC value
get_adc_value_ok:
    clr ADCDL                   ; disable data latch
    clr EOC                     ; Clearing read flag
    jmp loop                    ; for next data read
end
```

Body Fat Measurement Function

The body fat circuit consists of a sine generator, two operational amplifiers and a filter. It has high quality, high flexibility and high integration for body fat measurement. The whole module power is from LDO.

Sine Wave Generator

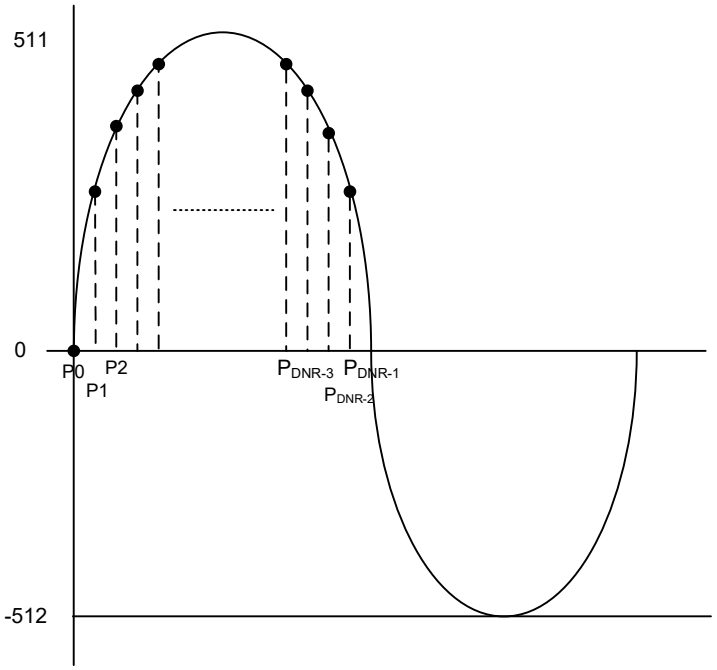
The sine generator consists of a frequency divider, a 5-bit counter, a Data Memory, a 10-bit D/A converter and the OP0. It offers a sine wave generator with a wide frequency range of 5kHz~500kHz and 32×9 bits of Data memory for sine wave pattern by software setting.The frequency divider will multiply by DN/M to generate a clock to counter. For related details refer to following formula.

- System clock/M = sine wave frequency
- System clock×(DN/M) = counter count rate
- The M must be the multiple of N and 8
- M = N×DN
- DNR = DN/2
- DN : The serial number of the sine wave cycle pattern (DN≤64)
- DNR : The serial number of the 1/2 sine wave cycle pattern stored in Data memory (DNR≤32)

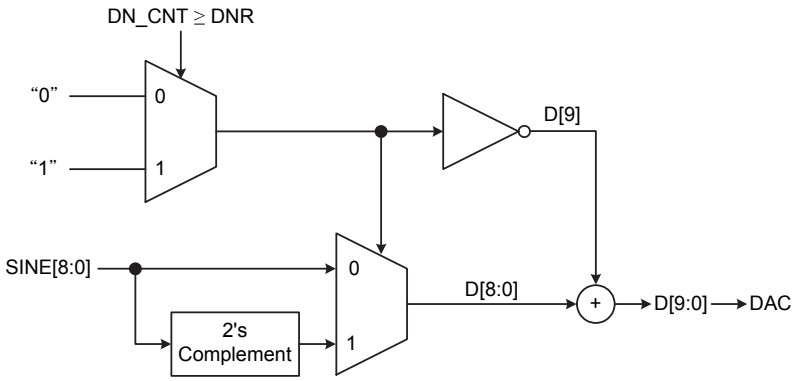
Please refer to following table and figure in details.

System Frequency	4MHz			8MHz		
Sine Wave Frequency (kHz)	500	50	5	500	50	5
M	8	80	800	16	160	1600
N	1	2	20	1	4	25
DN	8	40	40	16	40	64
DNR	4	20	20	8	20	32

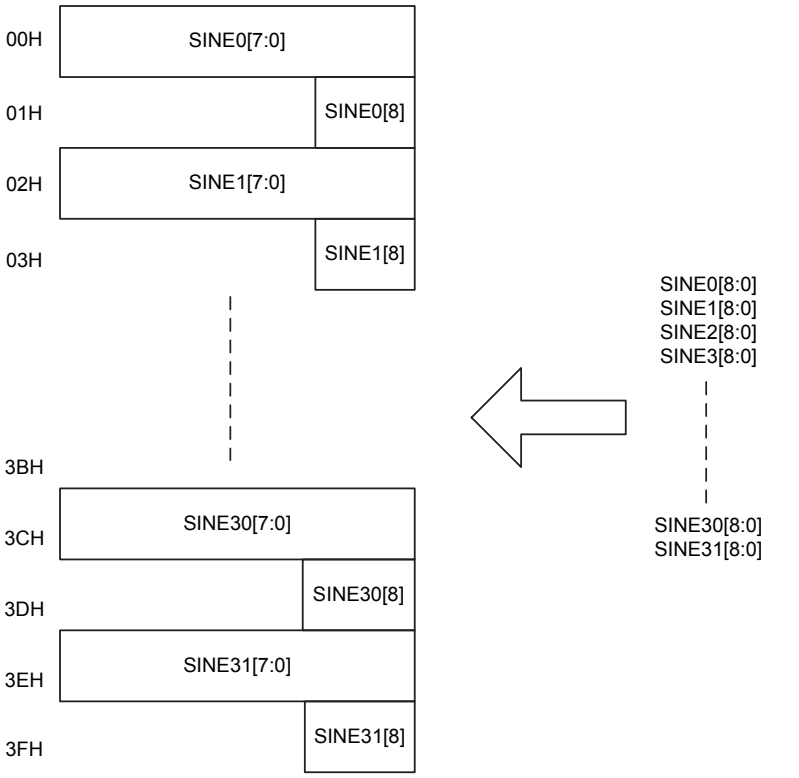
Note: The sine wave generator circuit consists of a 10-bit D/A converter and a smoothing filter whose frequency at -3dB is equal to 489kHz. When the output frequency is 500kHz, the output wave amplitude will decrease and therefore it is impossible to achieve a full range of V_{OREG}~0.



It is only necessary to generate a half sine wave pattern P_0~P_{DNR-1} which is stored in Data memory sector 3 (00H~3FH). The sine pattern SINEn[7:0] is stored using even addresses and the sine pattern SINEn[8] is stored using odd addresses. Once the sine generator is enabled, the CPU cannot write/read any data to/from this Data memory. The sine generator will then read data from this Data memory and transmit it to the 10-bit D/A converter. The controller reads a half sine pattern from Data memory and generates a sine waveform on the SIN pin. Refer to following figure.



Note: D[9] of DAC will be "1", when DN_CNT < DNR. And D[9] will be "0", when DN_CNT ≥ DNR.
It should be noted that one inverter is put ahead D[9].



Data Memory (Sector 3) **Sine Wave Pattern**

Body Fat Measurement Registers

There are a series of registers control the overall operation of the body fat measurement function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SGC	SGEN	D6	D5	BREN	—	—	—	—
SGN	—	—	D5	D4	D3	D2	D1	D0
SGDNR	—	—	—	D4	D3	D2	D1	D0
OPAC	OPAEN	—	—	—	OP2G3	OP2G2	OP2G1	OP2G0
SWC0	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
SWC1	—	—	—	—	—	—	SW9	SW8

Body Fat Measurement Register List

Sine Wave Generator

The sine wave generator is controlled by three registers. Details are given as below.

- **SGC Register**

Bit	7	6	5	4	3	2	1	0
Name	SGEN	D6	D5	BREN	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

Bit 7 **SGEN**: Sine generator control
0: Disable
1: Enable
When this bit is equal to "0", the OP0 and 10-bit D/A converter will be in a power down mode.

Bit 6~5 **D6~D5:** Reserved bits, must be fixed as "00"

Bit 4 **BREN**: Bias resistors control bit
0: Disable–power down mode
1: Enable–normal mode

Bit 3~0 Unimplemented, read as "0"

- **SGN Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	D5	D4	D3	D2	D1	D0
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 **D5~D0**: Sine generator data
The multiplicator of system frequency (N).
The multiplicator (N) is equal to D[5:0]+1.

• **SGDNR Register**

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

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **D4~D0**: Data number of sample

1/2 sine wave cycle numerical value is stored in Data memory Sector 3. DNR is equal to D[4:0]+1.

Operational Amplifier

Three registers are associated with the amplifier operation. The OPAC register is used for controlling the operational amplifier, the SWC0 and SWC1 registers are used for configuring the switch condition.

• **OPAC Register**

Bit	7	6	5	4	3	2	1	0
Name	OPAEN	—	—	—	OP2G3	OP2G2	OP2G1	OP2G0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **OPAEN**: Operational amplifier control
0: Disable
1: Enable
When this bit is equal to "0", OP1 and OP2 will be in a power down mode.

Bit 6~4 Unimplemented, read as "0"

Bit 3~0 **OP2G3~OP2G0**: OP2 gain control
0001: 1.14
0010: 1.31
0011: 1.50
0100: 1.73
0101: 2.00
0110: 2.33
0111: 2.75
1000: 3.285
1001: 4.00
1010: 5.00
Others: 1.00

• **SWC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
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 **SW7**: Switch 7 control bit
0: Off
1: On

Bit 6 **SW6**: Switch 6 control bit
0: Off
1: On

Bit 5 **SW5**: Switch 5 control bit
0: Off
1: On

Bit 4 **SW4**: Switch 4 control bit
0: Off
1: On

Bit 3 **SW3**: Switch 3 control bit
0: Off
1: On

Bit 2 **SW2**: Switch 2 control bit
0: Off
1: On

Bit 1 **SW1**: Switch 1 control bit
0: Off
1: On

Bit 0 **SW0**: Switch 0 control bit
0: Off
1: On

• **SWC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	SW9	SW8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1 **SW9**: Switch 9 control bit
0: Off
1: On

Bit 0 **SW8**: Switch 8 control bit
0: Off
1: On

Serial Peripheral Interface – SPI

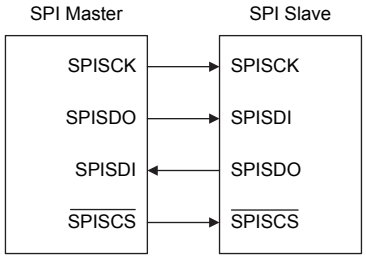
The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the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

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

SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SPISDI, SPISDO, SPISCK and $\overline{\text{SPISCS}}$. Pins SPISDI and SPISDO are the Serial Data Input and Serial Data Output lines, the SPISCK pin is the Serial Clock line and $\overline{\text{SPISCS}}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins, the SPI interface must first be enabled by configuring the corresponding selection bits in the pin-shared function selection registers. The SPI can be disabled or enabled using the SPIEN bit in the SPIC0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the devices only contain a single $\overline{\text{SPISCS}}$ pin only one slave device can be utilized.

The $\overline{\text{SPISCS}}$ pin is controlled by software, set SPICSEN bit to 1 to enable the $\overline{\text{SPISCS}}$ pin function, and clear SPICSEN bit to 0, the $\overline{\text{SPISCS}}$ pin will be floating state.

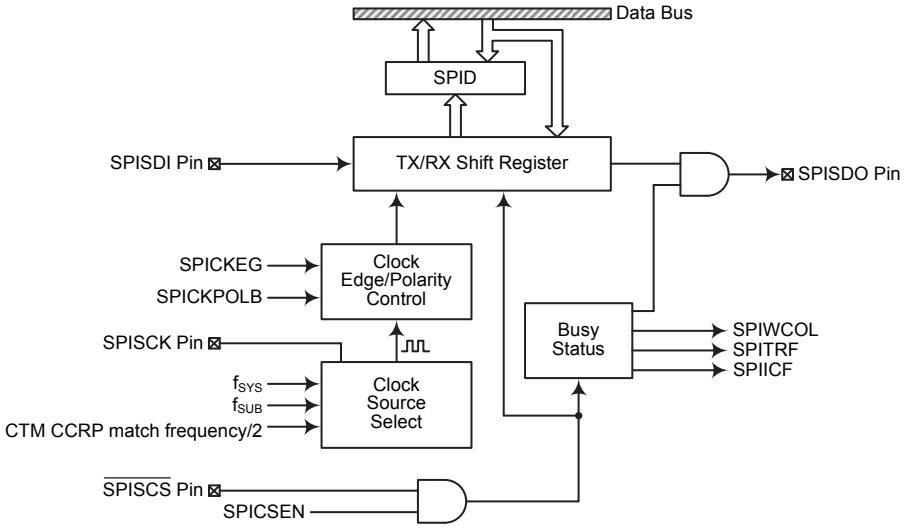


SPI Master/Slave Connection

The SPI function in the devices offer the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes
- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge

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



SPI Block Diagram

• SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SPID data register and two registers, SPIC0 and SPIC1.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SPIC0	SPIM2	SPIM1	SPIM0	—	—	—	SPIEN	SPIICF
SPIC1	—	—	SPICKPOLB	SPICKEG	SPIMLS	SPICSEN	SPIWCOL	SPITRF
SPID	D7	D6	D5	D4	D3	D2	D1	D0

SPI Register List

SPI Data Register

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

• SPID Register

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

"x": unknown

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

SPI Control Registers

There are also two control registers for the SPI interface, SPIC0 and SPIC1. The SPIC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SPIC1 register is used for other control functions such as LSB/MSB selection, write collision flag etc.

• **SPIC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SPIM2	SPIM1	SPIM0	—	—	—	SPIEN	SPIICF
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	1	1	1	—	—	—	0	0

Bit 7~5 **SPIM2~SPIM0**: SPI operating mode control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is CTM CCRP match frequency/2
101: SPI slave mode
110: SPI disable
111: SPI disable
These bits are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from CTM and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4~2 Unimplemented, read as "0"

Bit 1 **SPIEN**: SPI enable control
0: Disable
1: Enable
The bit is the overall on/off control for the SPI interface. When the SPIEN bit is cleared to zero to disable the SPI interface, the SPISDI, SPISDO, SPISCK and $\overline{\text{SPISCS}}$ lines will lose their SPI function and the SPI operating current will be reduced to a minimum value. When the bit is high the SPI interface is enabled.

Bit 0 **SPIICF:** SPI incomplete flag
0: SPI incomplete condition is not occurred
1: SPI incomplete condition is occured
This bit is only available when the SPI is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SPIEN and SPICSEN bits both being set high but the $\overline{\text{SPISCS}}$ line is pulled high by the external master device before the SPI data transfer is completely finished, the SPIICF bit will be set high together with the SPITRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the SPITRF bit will not be set high if the SPIICF bit is set high by software application program.

• **SPIC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SPICKPOLB	SPICKEG	SPIMLS	SPICSEN	SPIWCOL	SPITRF
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 **SPICKPOLB**: SPI clock line base condition selection
0: The SPISCK line will be high when the clock is inactive
1: The SPISCK line will be low when the clock is inactive
The SPICKPOLB bit determines the base condition of the clock line, if the bit is high, then the SPISCK line will be low when the clock is inactive. When the SPICKPOLB bit is low, then the SPISCK line will be high when the clock is inactive.

Bit 4 **SPICKEG**: SPI SPISCK clock active edge type selection

SPICKPOLB=0

0: SPISCK has high base level with data capture on SPISCK rising edge

1: SPISCK has high base level with data capture on SPISCK falling edge

SPICKPOLB=1

0: SPISCK has low base level with data capture on SPISCK falling edge

1: SPISCK has low base level with data capture on SPISCK rising edge

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

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

0: LSB first

1: MSB first

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

Bit 2 **SPICSEN**: SPI $\overline{\text{SPISCS}}$ pin control

0: Disable

1: Enable

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

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

0: No collision

1: Collision

The SPIWCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SPID register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared to zero by the application program.

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

0: SPI data is being transferred

1: SPI data transmission is completed

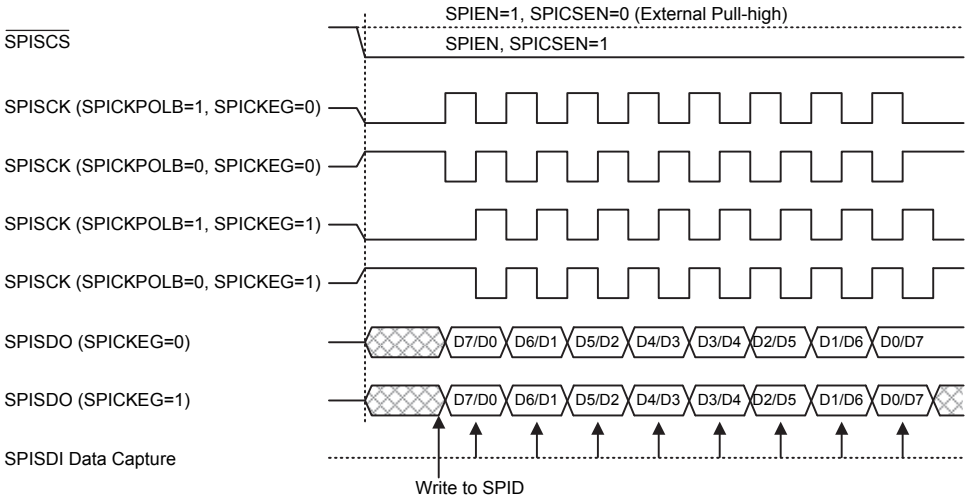
The SPITRF bit is the Transmit/Receive Complete flag and is set “1” automatically when an SPI data transmission is completed, but must set to zero by the application program. It can be used to generate an interrupt.

SPI Communication

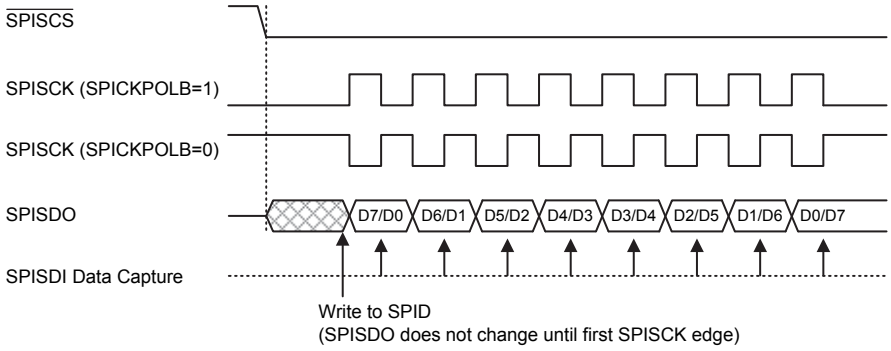
After the SPI interface is enabled by setting the SPIEN bit high, then in the Master Mode, when data is written to the SPID register, transmission/reception will begin simultaneously. When the data transfer is complete, the SPITRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SPID register will be transmitted and any data on the SPISDI pin will be shifted into the SPID register.

The master should output a $\overline{\text{SPISCS}}$ signal to enable the slave device before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the SPISCK signal depending upon the configurations of the SPICKPOLB bit and SPICKEG bit. The accompanying timing diagram shows the relationship between the slave data and SPISCK signal for various configurations of the SPICKPOLB and SPICKEG bits.

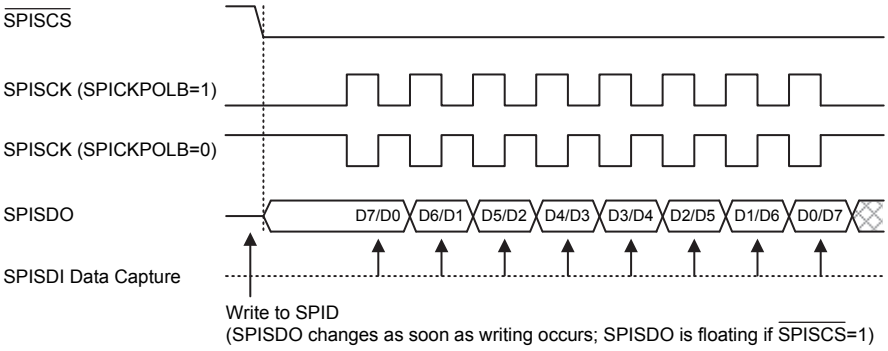
The SPI will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.



SPI Master Mode Timing

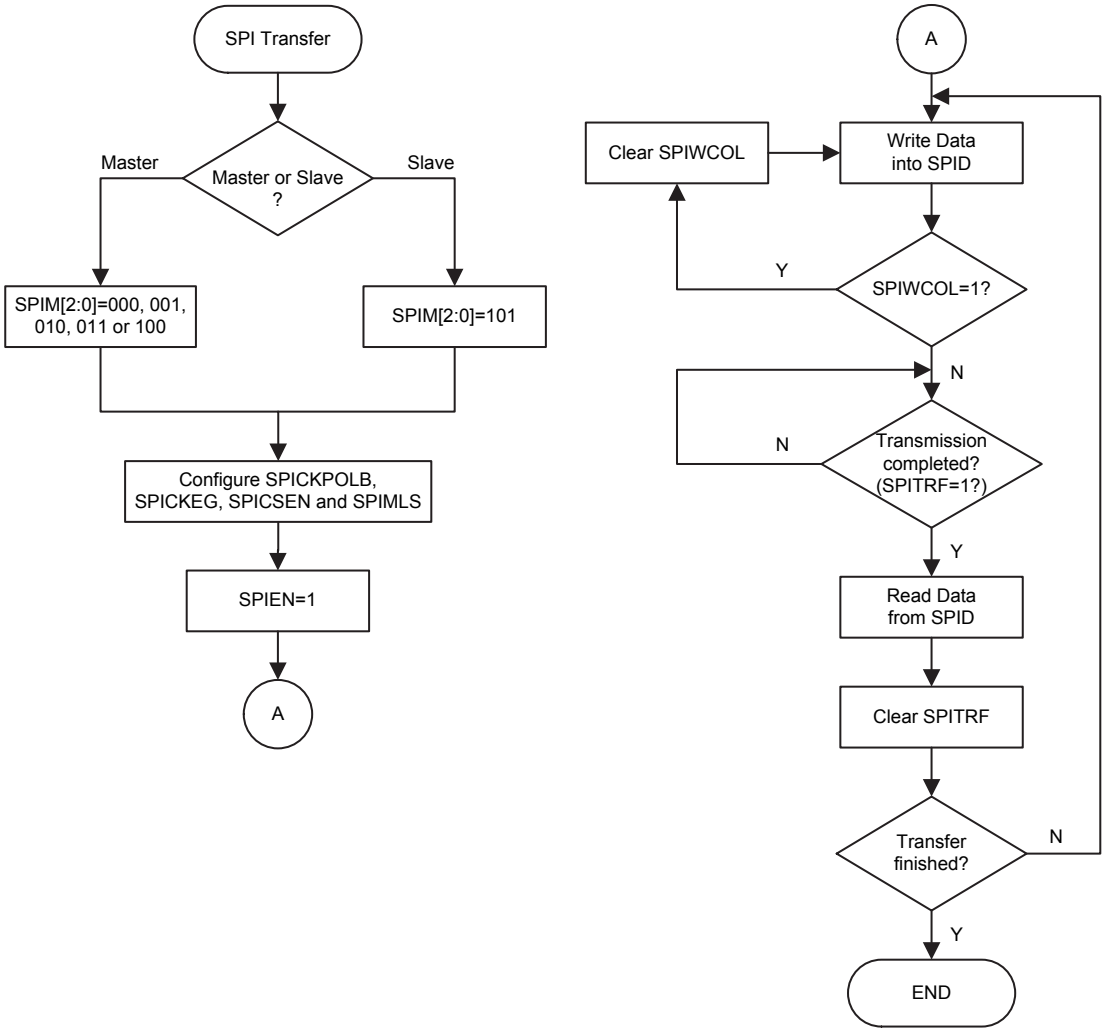


SPI Slave Mode Timing – SPICKEG=0



Note: For SPI slave mode, if SPIEN=1 and SPICSEN=0, SPI is always enabled and ignores the SPISCS level.

SPI Slave Mode Timing – SPICKEG=1



SPI Transfer Control Flowchart

SPI Bus Enable/Disable

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

When the SPI bus is diabled, SPISCK, SPISDI, SPISDO, $\overline{\text{SPISCS}}$ will become I/O pins or the other pin-shared functions by configuring the corresponding pin-shared control bits.

SPI Operation Steps

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

The SPICSEN bit in the SPIC1 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{\text{SPISCS}}$ line to be active, which can then be used to control the SPI interface. If the SPICSEN bit is low, the SPI interface will be disabled and the $\overline{\text{SPISCS}}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the SPICSEN bit and the SPIEN bit in the SPIC0 register are set high, this will place the SPISDI line in a floating condition and the SPISDO line high. If in Master Mode the SPISCK line will be either high or low depending upon the clock polarity selection bit SPICKPOLB in the SPIC1 register. If in Slave Mode the SPISCK line will be in a floating condition. If SPIEN is low then the bus will be disabled and $\overline{\text{SPISCS}}$, SPISDI, SPISDO and SPISCK will all become I/O pins or the other functions. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SPID register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode

- Step 1
  Select the SPI Master mode and clock source using the SPIM2~SPIM0 bits in the SPIC0 control register.
- Step 2
  Setup the SPICSEN bit and setup the SPIMLS bit to choose if the data is MSB or LSB first, this must be same as the Slave device.
- Step 3
  Setup the SPIEN bit in the SPIC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SPID register, which will actually place the data into the TXRX buffer. Then use the SPISCK and $\overline{\text{SPISCS}}$ lines to output the data. After this go to step 5.
  For read operations: the data transferred in on the SPISDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPID register.
- Step 5
  Check the SPIWCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the SPITRF bit or wait for a SPI serial bus interrupt.

- Step 7
  Read data from the SPID register.
- Step 8
  Clear SPITRF.
- Step 9
  Go to step 4.

Slave Mode

- Step 1
  Select the SPI Slave mode using the SPIM2~SPIM0 bits in the SPIC0 control register.
- Step 2
  Setup the SPICSEN bit and setup the SPIMLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master device.
- Step 3
  Setup the SPIEN bit in the SPIC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SPID register, which will actually place the data into the TXRX buffer. Then wait for the master clock SPISCK and $\overline{\text{SPISCS}}$ signal. After this, go to step 5.
  For read operations: the data transferred in on the SPISDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPID register.
- Step 5
  Check the SPIWCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the SPITRF bit or wait for a SPI serial bus interrupt.
- Step 7
  Read data from the SPID register.
- Step 8
  Clear SPITRF.
- Step 9
  Go to step 4.

Error Detection

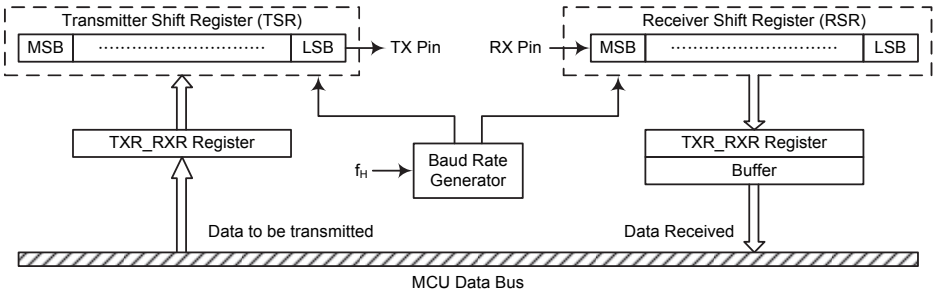
The SPIWCOL bit in the SPIC1 register is provided to indicate errors during data transfer. The bit is set by the SPI serial Interface but must be cleared by the application program. This bit indicates a data collision has occurred which happens if a write to the SPID register takes place during a data transfer operation and will prevent the write operation from continuing.

UART Interface

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

The integrated UART function contains the following features:

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



UART Data Transfer Block Diagram

UART External Pins

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

UART Data Transfer Scheme

The above block diagram shows the overall data transfer structure arrangement for the UART. The actual data to be transmitted from the MCU is first transferred to the TXR_RXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate Generator. Only the TXR_RXR register is mapped onto the MCU Data Memory, the Transmit Shift Register is not mapped and is therefore inaccessible to the application program.

Data to be received by the UART is accepted on the external RX pin, from where it is shifted in, LSB first, to the Receiver Shift Register at a rate controlled by the Baud Rate Generator. When the shift register is full, the data will then be transferred from the shift register to the internal TXR_RXR register, where it is buffered and can be manipulated by the application program. Only the TXR_RXR register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.

It should be noted that the actual register for data transmission and reception only exists as a single shared register in the Data Memory. This shared register known as the TXR_RXR register is used for both data transmission and data reception.

UART Status and Control Registers

There are five control registers associated with the UART function. The USR, UCR1 and UCR2 registers control the overall function of the UART, while the BRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXR data register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0

UART Register List

• **USR Register**

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

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

Bit 7 **PERR**: Parity error flag
0: No parity error is detected
1: Parity error is detected
The PERR flag is the parity error flag. When this read only flag is "0", it indicates a parity error has not been detected. When the flag is "1", it indicates that the parity of the received word is incorrect. This error flag is applicable only if Parity mode (odd or even) is selected. The flag can also be cleared to zero by a software sequence which involves a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 6 **NF**: Noise flag
0: No noise is detected
1: Noise is detected
The NF flag is the noise flag. When this read only flag is "0", it indicates no noise condition. When the flag is "1", it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of as overrun. The NF flag can be cleared to zero by a software sequence which will involve a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 5 **FERR**: Framing error flag
0: No framing error is detected
1: Framing error is detected
The FERR flag is the framing error flag. When this read only flag is "0", it indicates that there is no framing error. When the flag is "1", it indicates that a framing error has been detected for the current character. The flag can also be cleared to zero by a software sequence which will involve a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 4 **OERR**: Overrun error flag
0: No overrun error is detected
1: Overrun error is detected
The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is "0", it indicates that there is no overrun error. When the flag is "1", it indicates that an overrun error occurs which will inhibit further transfers to the TXR_RXR receive data register. The flag is cleared to zero by a software sequence, which is a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 3 **RIDLE**: Receiver status
0: Data reception is in progress (Data being received)
1: No data reception is in progress (Receiver is idle)
The RIDLE flag is the receiver status flag. When this read only flag is "0", it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is "1", it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is "1" indicating that the UART receiver is idle and the RX pin stays in logic high condition.

Bit 2 **RXIF**: Receive TXR_RXR data register status
0: TXR_RXR data register is empty
1: TXR_RXR data register has available data
The RXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the TXR_RXR read data register is empty. When the flag is "1", it indicates that the TXR_RXR read data register contains new data. When the contents of the shift register are transferred to the TXR_RXR register, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag is cleared when the USR register is read with RXIF set, followed by a read from the TXR_RXR register, and if the TXR_RXR register has no data available.

Bit 1 **TIDLE**: Transmission idle
0: Data transmission is in progress (Data being transmitted)
1: No data transmission is in progress (Transmitter is idle)
The TIDLE flag is known as the transmission complete flag. When this read only flag is "0", it indicates that a transmission is in progress. This flag will be set high when the TXIF flag is "1" and when there is no transmit data or break character being transmitted. When TIDLE is equal to "1", the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared to zero by reading the USR register with TIDLE set and then writing to the TXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

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

• UCR1 Register

The UCR1 register together with the UCR2 register are the two UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

"x": unknown

Bit 7 **UARTEN**: UART function enable control
0: Disable UART. TX and RX pins are in a floating state
1: Enable UART. TX and RX pins function as UART pins
The UARTEN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX pin as well as the TX pin will be set in a floating state. When the bit is equal to "1", the UART will be enabled and the TX and RX pins will function as defined by the TXEN and RXEN enable control bits.
When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits will be cleared to zero, while the

TIDLE, TXIF and RIDLE bits will be set high. Other control bits in UCR1, UCR2 and BRG registers will remain unaffected. If the UART is active and the UARTEN bit is cleared to zero, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

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

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

Bit 5 **PREN**: Parity function enable control
0: Parity function is disabled
1: Parity function is enabled

This is the parity enable bit. When this bit is equal to "1", the parity function will be enabled. If the bit is equal to "0", then the parity function will be disabled.

Bit 4 **PRT**: Parity type selection bit
0: Even parity for parity generator
1: Odd parity for parity generator

This bit is the parity type selection bit. When this bit is equal to "1", odd parity type will be selected. If the bit is equal to "0", then even parity type will be selected.

Bit 3 **STOPS**: Number of Stop bits selection
0: One stop bit format is used
1: Two stop bits format is used

This bit determines if one or two stop bits are to be used. When this bit is equal to "1", two stop bits are used. If this bit is equal to "0", then only one stop bit is used.

Bit 2 **TXBRK**: Transmit break character
0: No break character is transmitted
1: Break characters transmit

The TXBRK bit is the Transmit Break Character bit. When this bit is "0", there are no break characters and the TX pin operates normally. When the bit is "1", there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to "1", after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.

Bit 1 **RX8**: Receive data bit 8 for 9-bit data transfer format (read only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **TX8**: Transmit data bit 8 for 9-bit data transfer format (write only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UCR2 Register**

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

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

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

Bit 6 **RXEN**: UART Receiver enabled control
0: UART receiver is disabled
1: UART receiver is enabled
The bit named RXEN is the Receiver Enable Bit. When this bit is equal to "0", the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RX pin will be set in a floating state. If the RXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the receiver will be enabled and the RX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX pin will be set in a floating state.

Bit 5 **BRGH**: Baud Rate speed selection
0: Low speed baud rate
1: High speed baud rate
The bit named BRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register BRG, controls the Baud Rate of the UART. If this bit is equal to "1", the high speed mode is selected. If the bit is equal to "0", the low speed mode is selected.

Bit 4 **ADDEN**: Address detect function enable control
0: Address detect function is disabled
1: Address detect function is enabled
The bit named ADDEN is the address detect function enable control bit. When this bit is equal to "1", the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to RX7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the received word is "0" with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **WAKE**: RX pin wake-up UART function enable control
0: RX pin wake-up UART function is disabled
1: RX pin wake-up UART function is enabled
This bit is used to control the wake-up UART function when a falling edge on the RX

pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no RX pin wake-up UART function if the UART clock (f_H) exists. If the WAKE bit is set high as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the RX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the RX pin when the WAKE bit is cleared to zero.

Bit 2 **RIE**: Receiver interrupt enable control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled

This bit enables or disables the receiver interrupt. If this bit is equal to "1" and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.

Bit 1 **TIIE**: Transmitter Idle interrupt enable control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled

This bit enables or disables the transmitter idle interrupt. If this bit is equal to "1" and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

Bit 0 **TEIE**: Transmitter Empty interrupt enable control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled

This bit enables or disables the transmitter empty interrupt. If this bit is equal to "1" and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

• TXR_RXR Register

Bit	7	6	5	4	3	2	1	0
Name	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": unknown

Bit 7~0 **TXRX7~TXRX0**: UART Transmit/Receive Data bit 7 ~ bit 0

• BRG Register

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

"x": unknown

Bit 7~0 **D7~D0**: Baud Rate values

By programming the BRGH bit in UCR2 Register which allows selection of the related formula described above and programming the required value in the BRG register, the required baud rate can be setup.

Note: Baud rate=f_H/[64×(N+1)] if BRGH=0.
Baud rate=f_H/[16×(N+1)] if BRGH=1.

Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the baud rate register BRG and the second is the value of the BRGH bit with the control register UCR2. The BRGH bit decides if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value N in the BRG register which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the BRG register and has a range of between 0 and 255.

UCR2 BRGH Bit	0	1
Baud Rate (BR)	f_H / [64 (N+1)]	f_H / [16 (N+1)]

By programming the BRGH bit which allows selection of the related formula and programming the required value in the BRG register, the required baud rate can be setup. Note that because the actual baud rate is determined using a discrete value, N, placed in the BRG register, there will be an error associated between the actual and requested value. The following example shows how the BRG register value N and the error value can be calculated.

Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, and with BRGH cleared to zero determine the BRG register value N, the actual baud rate and the error value for a desired baud rate of 4800.

From the above table the desired baud rate $BR=f_H/[64(N+1)]$

Re-arranging this equation gives $N=[f_H/(BR\times64)]-1$

Giving a value for $N=[4000000/(4800\times64)]-1=12.0208$

To obtain the closest value, a decimal value of 12 should be placed into the BRG register. This gives an actual or calculated baud rate value of $BR=4000000/[64\times(12+1)]=4808$

Therefore the error is equal to (4808-4800)/4800=0.16%

UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding BNO, PRT, PREN, and STOPS bits in the UCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

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

Clearing the UARTEN bit will disable the TX and RX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same

time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2 and BRG registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

Data, Parity and Stop Bit Selection

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

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

Transmitter Receiver Data Format

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

8-bit data format

9-bit data format

UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR_RXR register. The data to be transmitted is loaded into this TXR_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the TXR_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program

for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN bit is set, but the data will not be transmitted until the TXR_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR_RXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

Transmitting Data

When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant bit first. In the transmit mode, the TXR_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8 bit in the UCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT, PREN and STOPS bits to define the required word length, parity type and number of stop bits.
- Setup the BRG register to select the desired baud rate.
- Set the TXEN bit to ensure that the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR_RXR register. Note that this step will clear the TXIF bit.

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

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

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

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

During a data transmission, a write instruction to the TXR_RXR register will place the data into the TXR_RXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLE bit the following software sequence is used:

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

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

Transmit Break

If the TXBRK bit is set then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13×N '0' bits and stop bits, where N=1, 2, etc. If a break character is to be transmitted then the TXBRK bit must be first set by the application program, and then cleared to generate the stop bits. Transmitting a break character will not generate a transmit interrupt. Note that a break condition length is at least 13 bits long. If the TXBRK bit is continually kept at a logic high level then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the TXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out one or two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the BNO bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8 bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

Receiving Data

When the UART receiver is receiving data, the data is serially shifted in on the external RX input pin, LSB first. In the read mode, the TXR_RXR register forms a buffer between the internal bus and the receiver shift register. The TXR_RXR register is a two byte deep FIFO data buffer, where two bytes can be held in the FIFO while a third byte can continue to be received. Note that the application program must ensure that the data is read from TXR_RXR before the third byte has been completely shifted in, otherwise this third byte will be discarded and an overrun error OERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNO, PRT and PREN bits to define the word length, parity type.
- Setup the BRG register to select the desired baud rate.
- Set the RXEN bit to ensure that the RX pin is used as a UART receiver pin.

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

When a character is received the following sequence of events will occur:

- The RXIF bit in the USR register will be set when the TXR_RXR register has data available. There will be at most one more character available before an overrun error occurs.
- When the contents of the shift register have been transferred to the TXR_RXR register, then if the RIE bit is set, an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

The RXIF bit can be cleared using the following software sequence:

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

Receive Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the BNO bit plus one stop bit. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO plus one stop bit. The RXIF bit is set, FERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLE bit is set. A break is regarded as a character that contains only zeros with the FERR flag set. If a long break signal has been detected, the receiver will regard it as a data frame including a start bit, data bits and the invalid stop bit and the FERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

- The framing error flag, FERR, will be set.
- The receive data register, TXR_RXR, will be cleared.
- The OERR, NF, PERR, RIDLE or RXIF flags will possibly be set.

Idle Status

When the receiver is reading data, which means it will be in between the detection of a start bit and the reading of a stop bit, the receiver status flag in the USR register, otherwise known as the RIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the RIDLE flag will have a high value, which indicates the receiver is in an idle condition.

Receiver Interrupt

The read only receive interrupt flag RXIF in the USR register is set by an edge generated by the receiver. An interrupt is generated if RIE=1, when a word is transferred from the Receive Shift Register, RSR, to the Receive Data Register, TXR_RXR. An overrun error can also generate an interrupt if RIE=1.

Managing Receiver Errors

Several types of reception errors can occur within the UART module, the following section describes the various types and how they are managed by the UART.

Overrun Error – OERR

The TXR_RXR register is composed of a two byte deep FIFO data buffer, where two bytes can be held in the FIFO register, while a third byte can continue to be received. Before this third byte has been entirely shifted in, the data should be read from the TXR_RXR register. If this is not done, the overrun error flag OERR will be consequently indicated.

In the event of an overrun error occurring, the following will happen:

- The OERR flag in the USR register will be set.
- The TXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIE bit is set.

The OERR flag can be cleared by an access to the USR register followed by a read to the TXR_RXR register.

Noise Error – NF

Over-sampling is used for data recovery to identify valid incoming data and noise. If noise is detected within a frame the following will occur:

- The read only noise flag, NF, in the USR register will be set on the rising edge of the RXIF bit.
- Data will be transferred from the Shift register to the TXR_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the RXIF bit which itself generates an interrupt.

Note that the NF flag is reset by a USR register read operation followed by a TXR_RXR register read operation.

Framing Error – FERR

The read only framing error flag, FERR, in the USR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, both stop bits must be high; otherwise the FERR flag will be set. The FERR flag and the received data will be recorded in the USR and TXR_RXR registers respectively, and the flag is cleared in any reset.

Parity Error – PERR

The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd or even is selected. The read only PERR flag and the received data will be recorded in the USR and TXR_RXR registers respectively. It is cleared on any reset, it should be noted that the flags, FERR and PERR, in the USR register should first be read by the application program before reading the data word.

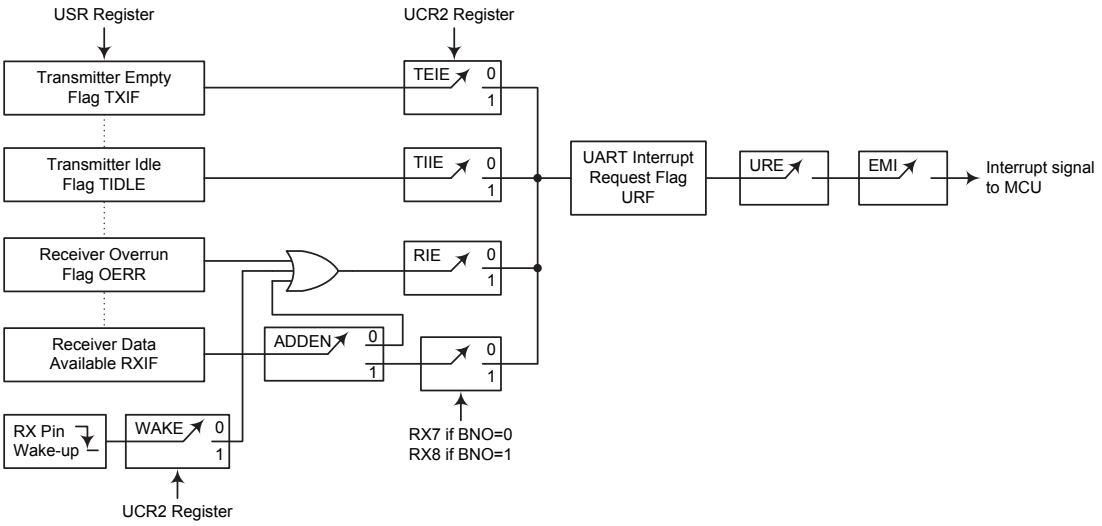
UART Interrupt Structure

Several individual UART conditions can trigger a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. When any of these conditions are created, if the global interrupt enable bit, the UART interrupt control bit is enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding USR register flags which will generate a UART interrupt if its associated interrupt enable control bit in the UCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual UART interrupt sources.

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the UART clock (f_H) source is switched off and the WAKE and RIE bits in the UCR2 register are set when a falling edge on the RX pin occurs. Note that in the event of an RX wake-up interrupt occurring, there will be a certain period of delay, commonly known as the System Start-up Time, for the oscillator to restart and stabilize before the system resumes normal operation.

Note that the USR register flags are read only and cannot be cleared or set by the application program, neither will they be cleared when the program jumps to the corresponding interrupt servicing routine, as is the case for some of the other interrupts. The flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the

UART register section. The overall UART interrupt can be disabled or enabled by the related interrupt enable control bits in the interrupt control registers of the microcontroller to decide whether the interrupt requested by the UART module is masked out or allowed.



UART Interrupt Structure

Address Detect Mode

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the URE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to zero.

ADDEN	Bit 9 if BNO=1 Bit 8 if BNO=0	UART Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

ADDEN Bit Function

UART Power Down and Wake-up

When the UART clock (f_H) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock (f_H) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the IDLE or SLEEP mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note

that the USR, UCR1, UCR2, transmit and receive registers, as well as the BRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the UART clock (f_H) is off, then a falling edge on the RX pin will trigger an RX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX pin will be ignored

For a UART wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URE, must be set. If the EMI and URE bits are not set then only a wake up event will occur and no interrupt will be generated. Note also that as it takes certain system clock cycles after a wake-up before normal microcontroller resumes, the UART interrupt will not be generated until after this time has elapsed.

Low Voltage Detector – LVD

The devices have a Low Voltage Detector function, also known as LVD. This enables the devices to monitor the power supply voltage, V_{DD}, or the LVDIN 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 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	VBGEN	VLVD2	VLVD1	VLVD0
R/W	—	—	R	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 **LVDO**: LVD output flag
0: No low voltage detected
1: Low voltage detected

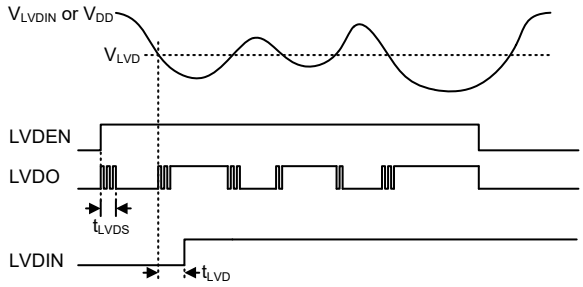
Bit 4 **LVDEN**: Low Voltage Detector enable control
0: Disable
1: Enable

Bit 3 **VBGEN**: Bandgap voltage output enable control
0: Disable
1: Enable
Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VBGEN bit is set high.

Bit 2~0 **VLVD2~VLVD0**: LVD voltage selection
000: $V_{LVDIN} \leq 1.04V$
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V
When the VLVD bit field is set to 000B, the LVD function operates by comparing the LVD reference voltage with the LVDIN pin input voltage. Otherwise, the LVD function operates by comparing the LVD reference voltage with the power supply voltage when the VLVD bit field is set to any other value except 000B.

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, or the LVDIN input voltage, with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.04V and 4.0V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. When the device is in the SLEEP mode, the low voltage detector will be disabled even if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} or V_{LVDIN} 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 has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} or V_{LVDIN} 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.

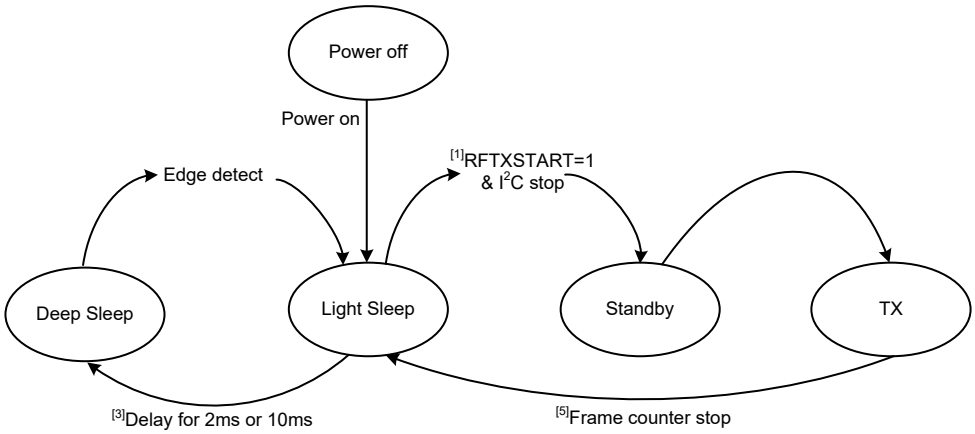
2.4GHz RF Transmitter

The RF portion is a fully integrated 2.4GHz transmitter. It consists of a fractional-N synthesizer, a programmable power amplifier (PA) and power management. The synthesizer loop filter, 4.8GHz VCO and Digital Controlled XO (DCXO) are all integrated.

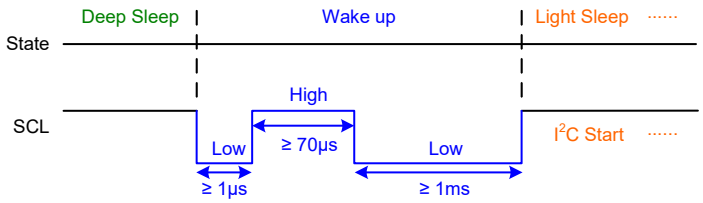
The transmitting session is an enhanced VCO direct modulation architecture. The GFSK modulation signal is fed into the PLL loop to take advantage of the fractional-N synthesizer. Another signal path is fed into the VCO directly to compensate the PLL loop response. As a result, it can generate a low FSK error GFSK signal. The modulated signal is fed into a power amplifier (PA) and the maximum output power can be up to +8dBm.

State Machine

The device provides five operating modes, Power Off mode, Deep Sleep mode, Light Sleep mode, Standby mode and TX mode. User can set RF parameters, transmit data and wake up the RF chip via the I²C interface.

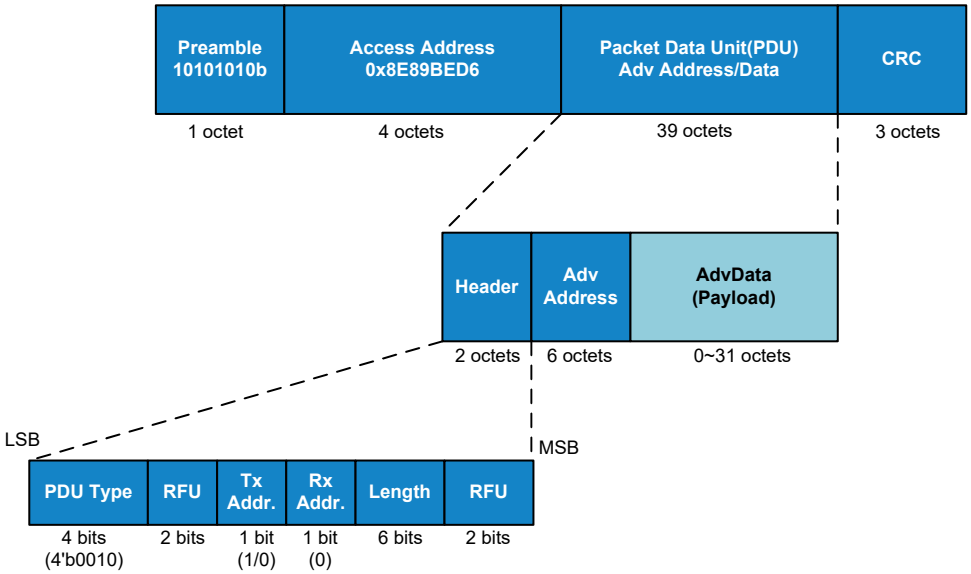


Note: 1. Once the RFTXSTART control bit is enabled, the I²C will stop and the RF chip will start transmiting data.
2. Deep Sleep mode: X'tal off.
3. Light Sleep mode: X'tal on, RF frequency synthesizer off.
4. Standby mode: X'tal on, RF frequency synthesizer on, RF PA off.
5. TX mode: X'tal on, RF frequency synthesizer on, RF PA on.
6. In the Light Sleep mode, if the SDA and SCL pin states both keep unchanged for 10ms, the device state will change to the Deep Sleep mode. If the SDA or SCL pin state has been toggled and then keeps unchanged, the timer will reset and recount until the 10ms time is up and then enter the Deep Sleep mode.
7. The device will be woken up from the Deep Sleep mode if a falling edge is detected on the SCL pin and the low pulse width should be maintained at least 1ms to return to Light Sleep state. After this, the master MCU can control the device based on the I²C format.

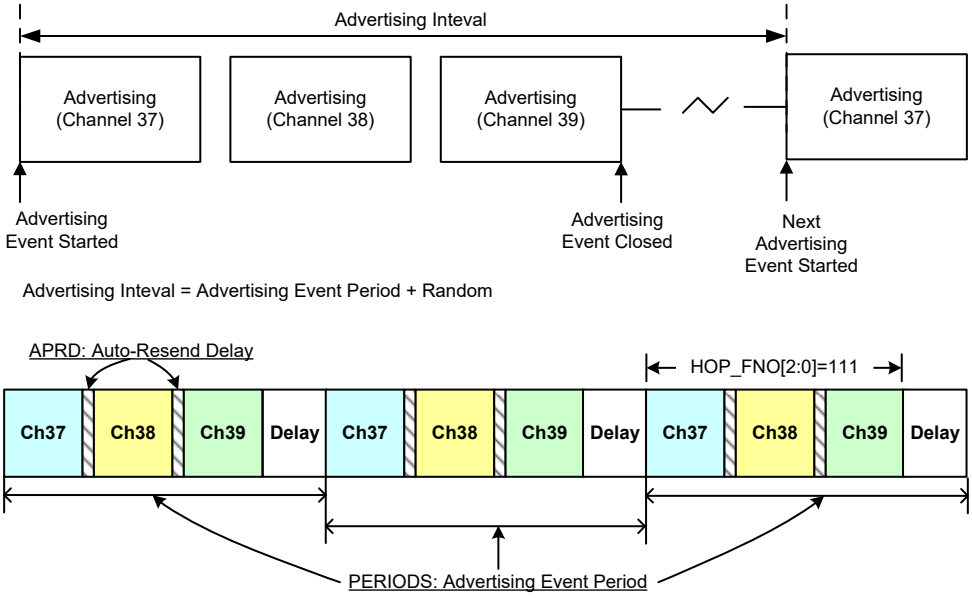


8. Each frame will be sent consecutively until the frame counter (PKT_AUTORS) is finished then enter the Light Sleep mode.

Packet Format



Packet Event Timing



PKT_APRD[4:0]: Packet format auto-resend delay, 250μs (Min.) ~ 8ms (Max.)

PKT_AUTORS[7:0]: Packet event auto-resend times

0: No resend (auto-resend disabled)

1: Resend 1 time

2: Resend 2 times

:

254: Resend 254 times

255: Always resend periodicly until the MCU forcibly turns off the TX transmitter.

During the TX transmission, if the MCU clears the RFTXSTART bit to zero, the TX transmitter will stop after the current advertising event is completed.

PKT_PERIODS[9:0]: Advertising event period

Period=10ms×(1+PKT_PERIODS[9:0])

00-0000-0000: 10ms

00-0000-0001: 20ms

…

11-1111-1111: 10240ms (10.24s)

HOP_FNO[2:0]: Hopping frequency number

Bit 0/Bit 1/Bit 2 indicates the enable bit for Ch37/Ch38/Ch39 respectively

It is not recommended to set these bits to "000". Ensure that there is at least one channel enabled.

For example:

001: Ch37 enabled, Ch38/Ch39 disabled; each advertising event only includes Ch37

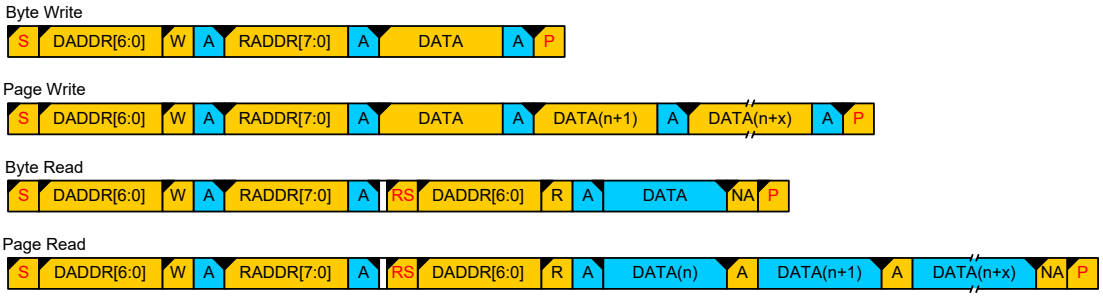
110: Ch38/Ch39 enabled, Ch37 disabled; each advertising event only includes Ch38 and Ch39

101: Ch37/Ch39 enabled, Ch38 disabled; each advertising event only includes Ch37 and Ch39

111: Ch37/Ch38/Ch39 enabled; each advertising event includes Ch37, Ch38 and Ch39.

I²C Serial Programming

The device supports the I²C format for byte write, page write, byte read and page read formats. Regarding the page write/read, register address will not automatically increase for the FIFO port (address: 10h) when continuously writing/reading data to/from the FIFO.



Bus Direction: : Host to device; : Device to host

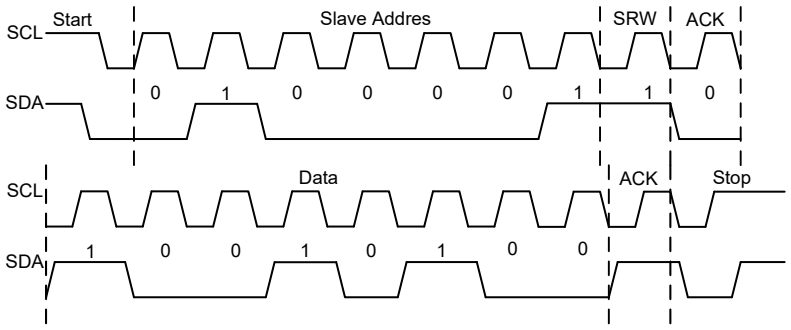
Symbol Definitions: S: Start; RS: Repeated Start; P: Stop

DADDR[6:0]: Device Address, 71h

R/W: Read/Write select; R: Read(1); W: Write(0)

RADDR[7:0]: register address

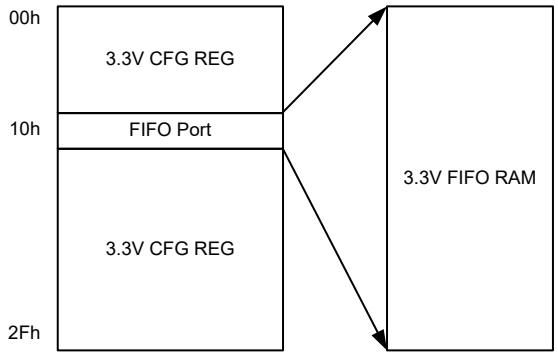
A: ACK(0); NA: NAK(1)



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

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

Memory Mapping



The steps for writing data to the FIFO via the I²C are described below:

1. Set the RSTPDUFF bit to 1 and clear the FIFO pointer. When the FIFO has been cleared the RSTPDUFF bit will be cleared to zero automatically.
2. Write data continuously to the address 10h. The I²C register address will stay at 10h without increament.
3. Configure the PDULEN bit field with a PDU data length to be transmitted.
4. To re-configure the PDU data, repeat step 1~3.

Configuration Registers

Addr.	Name	Bit							
		7	6	5	4	3	2	1	0
00h	CFG0	—		XO_TRIM[5:0]					
07h	CFG7	Setting1							
0Ch	CFGC	RFTXP_1							
0Dh	CFGD	RFTXP_2							
10h	CFG10	PDUDATA[7:0]							
11h	CFG11	RSTPDUFF	PDULEN[6:0]						
12h	CFG12	—		PDUDPTR[5:0]					

Addr.	Name	Bit							
		7	6	5	4	3	2	1	0
15h	CFG15	PKT_AUTORS[7:0]							
16h	CFG16	APRD_PDTH[1:0]		RNDDLY_EN	PKT_APRD[4:0]				
17h	CFG17	PKT_PERIODS[7:0]							
18h	CFG18	—						PKT_PERIODS[9:8]	
1Ah	CFG1A	RFTXSTART	—				HOP_FNO[2:0]		
1Dh	CFG1D	Cal1	—						
1Eh	CFG1E	—							Cal2
25h	CFG25	—	GIO3S[2:0]			—			
26h	CFG26	—				GIOPU	—		
27h	CFG27	—	LSTOS	LSTOM[1:0]		—			
2Ah	CFG2A	Setting2							
30h	CFG30	CHIPID[7:0]							
31h	CFG31	CHIPID[15:8]							
33h	CFG33	RMSOUT	—						RST_RF
36h	CFG36	—							TX_FLAG

Note that for the addresses which are not listed in this table, it is suggested not to change their initial values.

- **CFG0: Configuration Register 0**

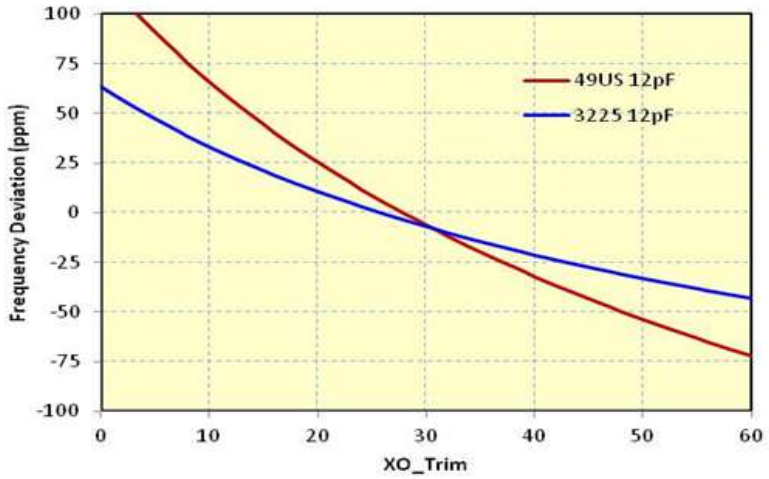
Bit	7	6	5	4	3	2	1	0
Name	—	—	XO_TRIM[5:0]					
R/W	—	—	R/W					
POR	0	0	1	0	0	0	0	0

Bit 7~6 Reserved bits, cannot be changed

Bit 5~0 **XO_TRIM[5:0]**: Trim value for the internal capacitor load of the crystal

49US 32MHz XO with a 12pF C_{LOAD}: The seggested setting is 1CH. Within ±25ppm frequency error, 1 trim code shifts -2.95ppm.

3225SMD 32MHz XO with a 12pF C_{LOAD}: The seggested setting is 1AH. Within ±25ppm frequency error, 1 trim code shifts -1.75ppm.



• **CFG7: Configuration Register 7**

Bit	7	6	5	4	3	2	1	0
Name	Setting1							
R/W	R/W							
POR	1	0	0	1	1	0	0	1

Bit 7~0 Reserved bits, must be set to "10010101" after power on

• **CFGC: Configuration Register C**

Bit	7	6	5	4	3	2	1	0
Name	RFTXP_1							
R/W	R/W							
POR	0	0	1	0	0	0	0	1

• **CFGD: Configuration Register D**

Bit	7	6	5	4	3	2	1	0
Name	RFTXP_2							
R/W	R/W							
POR	1	0	0	0	0	1	1	1

The recommended setting values (hexadecimal) for the CFGC and CFGD registers are listed below.

TX Power	High Power Matching		Low Power Matching	
	CFGC (RFTXP_1)	CFGD (RFTXP_2)	CFGC (RFTXP_1)	CFGD (RFTXP_2)
8dBm	A1	AF		
5dBm	A2	67	A1	A7
2dBm			A2	A1
0dBm			AF	D7
-2dBm	AF	D7		
-5dBm	AF	77	AF	73
-10dBm	AF	71	AF	71

• **CFG10: Configuration Register 10**

Bit	7	6	5	4	3	2	1	0
Name	PDUDATA[7:0]							
R/W	W							
POR	x	x	x	x	x	x	x	x

Bit 7~0 **PDUDATA[7:0]**: PDU data FIFO write port

• **CFG11: Configuration Register 11**

Bit	7	6	5	4	3	2	1	0
Name	RSTPDUFF	PDULEN[6:0]						
R/W	R/W	R/W						
POR	0	0	0	1	1	1	1	1

Bit 7 **RSTPDUFF**: Reset PDU data FIFO, automatically cleared after completion

Bit 6~0 **PDULEN[6:0]**: PDU data length (unit: octet); maximum: 39 octets

• **CFG12: Configuration Register 12**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PDUDPTR[5:0]					
R/W	—	—	R/W					
POR	0	0	0	0	0	0	0	0

Bit 7~6 Reserved bits, cannot be changed

Bit 5~0 **PDUDPTR[5:0]**: PDU data pointer, points to the start address to execute CRC/ whitening operations in PDU FIFO

• **CFG15: Configuration Register 15**

Bit	7	6	5	4	3	2	1	0
Name	PKT_AUTORS[7:0]							
R/W	R/W							
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PKT_AUTORS[7:0]**: Packet event auto-resend times
00h: No resend, auto-resend disabled
01h: Resend 1 time
02h: Resend 2 times
...
FFh: Always resend until RFTXSTART=0

• **CFG16: Configuration Register 16**

Bit	7	6	5	4	3	2	1	0
Name	APRD_PDTH[1:0]		RNDDLY_EN	PKT_APRD[4:0]				
R/W	R/W		R/W	R/W				
POR	0	0	1	1	1	1	1	1

Bit 7~6 **APRD_PDTH[1:0]**: Auto power down threshold
00: 1ms
01: 1.5ms
10: 2ms
11: 3ms
These bits define the time threshold that the device should automatically power down, i.e. enter the Deep Sleep mode. The device will not enter the Deep Sleep mode if the automatic retransmission interval defined by PKT_APRD[4:0] is less than the automatic power down threshold defined by APRD_PDTH[1:0].

Bit 5 **RNDDLY_EN**: Enable random delay (250~8000μs) per advertising event period
0: Disable
1: Enable

Bit 4~0 **PKT_APRD[4:0]**: Packet format auto-resend delay
00000: 250μs
00001: 500μs
00010: 750μs
…
11111: 8000μs (8ms)

• **CFG17: Configuration Register 17**

Bit	7	6	5	4	3	2	1	0
Name	PKT_PERIODS[7:0]							
R/W	R/W							
POR	0	1	1	0	0	0	1	1

Bit 7~0 **PKT_PERIODS[7:0]**: Advertising event period low byte

• **CFG18: Configuration Register 18**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PKT_PERIODS[9:8]	
R/W	—	—	—	—	—	—	R/W	
POR	0	0	0	0	0	0	0	0

Bit 7~2 Reserved bits, cannot be changed

Bit 1~0 **PKT_PERIODS[9:8]**: Advertising event period high byte
Delay=10ms×(1+PKT_PERIODS[9:0]); range: 10ms (000h) ~ 10240ms (3FFh)

• **CFG1A: Configuration Register 1A**

Bit	7	6	5	4	3	2	1	0
Name	RFTXSTART	—	—	—	—	HOP_FNO[2:0]		
R/W	R/W	—	—	—	—	R/W		
POR	0	0	0	0	0	1	1	1

Bit 7 **RFTXSTART**: RF layer2 transmission start control
0: RF layer2 transmission stops
1: RF layer2 transmission starts
This bit will be cleared to zero by hardware when the automatic retransmission count is full and all the payloads are completed. If this bit is cleared by the MCU, the RF TX will stop transmission.
Note: When the RFTXSTART bit changes from 1 to 0, which means to finish the WOT mode, the device needs an additional 70μs to exit the WOT mode. During this time any I^2C commands including WAKEUP would not be accepted.

Bit 6~3 Reserved bits, cannot be changed

Bit 2~0 **HOP_FNO[2:0]**: Hopping frequency number
HOP_FNO[0]: Enable channel 37 (2402MHz)
HOP_FNO[1]: Enable channel 38 (2426MHz)
HOP_FNO[2]: Enable channel 39 (2480MHz)
It is not recommended to set these bits to “000”. Ensure that there is at least one channel enabled.

• **CFG1D: Configuration Register 1D**

Bit	7	6	5	4	3	2	1	0
Name	Cal1	—	—	—	—	—	—	—
R/W	R/W	—	—	—	—	—	—	—
POR	0	0	0	1	1	1	1	0

Bit 7 **Cal1**: Described in the CFG1E register

Bit 6~0 Reserved bits, cannot be changed

• **CFG1E: Configuration Register 1E**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	Cal2
R/W	—	—	—	—	—	—	—	R/W
POR	0	0	1	1	0	1	0	0

Bit 7~1 Reserved bits, cannot be changed

Bit 0 **Cal2**: Cal1 and Cal2 both should be set to "1" during every power on, then wait until both are cleared to "0", which means IC calibration has finished.

• **CFG25: Configuration Register 25**

Bit	7	6	5	4	3	2	1	0
Name	—	GIO3S[2:0]			—	—	—	—
R/W	—	R/W			—	—	—	—
POR	0	0	0	0	0	0	0	0

Bit 7 Reserved bit, cannot be changed

Bit 6~4 **GIO3S[2:0]**: GIO3 pin function selection
000: No function, input
001~100: Reserved
101: RFTXSTART output
110: Reserved
111: TXACT, output

Bit 3~0 Reserved bits, cannot be changed

• **CFG26: Configuration Register 26**

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

Bit 7~4 Reserved bits, cannot be changed

Bit 3 **GIOPU**: GIO3 pull-up resistor control
0: Disable
1: Enable

Bit 2~0 Reserved bits, cannot be changed

• **CFG27: Configuration Register 27**

Bit	7	6	5	4	3	2	1	0
Name	—	LSTOS	LSTOM[1:0]		—	—	—	—
R/W	—	R/W	R/W		—	—	—	—
POR	1	0	0	0	0	0	0	0

Bit 7 Reserved bit, cannot be changed

Bit 6 **LSTOS**: Light sleep time-out selection
0: 2ms
1: 10ms

Bit 5~4 **LSTOM[1:0]**: Light sleep time-out mode selection
00: I^2C time-out turned on with 2ms/10ms delay, TX time-out turned on with 0s delay
01/10: I^2C time-out turned on with 2ms/10ms delay, TX time-out turned on with 2ms/10ms delay
11: Time-out off, always in light sleep mode

Bit 3~0 Reserved bits, cannot be changed

• **CFG2A: Configuration Register 2A**

Bit	7	6	5	4	3	2	1	0
Name	Setting2							
R/W	R/W							
POR	0	1	0	1	0	0	0	0

Bit 7~0 Reserved, must be set to “01000111” after power on

• **CFG30: Configuration Register 30**

Bit	7	6	5	4	3	2	1	0
Name	CHIPID[7:0]							
R/W	R							
POR	0	1	1	0	0	0	0	1

Bit 7~0 **CHIPID[7:0]**: Chip ID low byte, 0x61

• **CFG31: Configuration Register 31**

Bit	7	6	5	4	3	2	1	0
Name	CHIPID[15:8]							
R/W	R							
POR	0	1	1	1	0	0	0	1

Bit 7~0 **CHIPID[15:8]**: Chip ID high byte, 0x71

• **CFG33: Configuration Register 33**

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

Bit 7 **RMSOUT**: VCO oscillation detection (read only)
When the VCO oscillates and becomes stable, the RMSOUT bit will be set high by hardware. Otherwise the bit will be cleared to zero.

Bit 6~1 Reserved bits, cannot be changed

Bit 0 **RST_RF**: Registers reset control
Setting this bit high will reset the registers in the addresses 30h~3Fh. This bit will be cleared to zero after the reset is completed.

• **CFG36: Configuration Register 36**

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

Bit 7~1 Reserved bits, cannot be changed

Bit 0 **TX_FLAG**: TX status flag
0: TX is free
1: TX is busy

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The devices contain several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, LVD and the A/D converter, 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~MFI2 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pin	INTnE	INTnF	n=0~1
A/D Converter	ADE	ADF	—
SPI	SPIE	SPIF	—
Time Base	TBnE	TBnF	n=0~1
UART	URE	URF	—
Multi-function	MFnE	MFnF	n=0~2
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
CTM	CTMPE	CTMPF	—
	CTMAE	CTMAF	—
STM	STMPE	STMPF	—
	STMAE	STMAF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
INTC1 (BH66F71652)	SPIF	MF2F	—	MF0F	SPIE	MF2E	—	MF0E
INTC1 (BH66F71662)	SPIF	MF2F	MF1F	MF0F	SPIE	MF2E	MF1E	MF0E
INTC2	—	URF	TB1F	TB0F	—	URE	TB1E	TB0E
MFI0	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE
MFI1 (BH66F71662)	—	—	STMAF	STMPF	—	—	STMAE	STMPE
MFI2	—	—	DEF	LVF	—	—	DEE	LVE

Interrupt Register List

- **INTEG Register**

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

Bit 7~4 Unimplemented, read as "0"

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

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

- **INTC0 Register**

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

Bit 7 Unimplemented, read as "0"

Bit 6 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

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

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

Bit 3 **ADE**: A/D Converter interrupt control
0: Disable
1: Enable

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

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

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

• **INTC1 Register – BH66F71652**

Bit	7	6	5	4	3	2	1	0
Name	SPIF	MF2F	—	MF0F	SPIE	MF2E	—	MF0E
R/W	R/W	R/W	—	R/W	R/W	R/W	—	R/W
POR	0	0	—	0	0	0	—	0

Bit 7 **SPIF**: SPI interrupt request flag
0: No request
1: Interrupt request

Bit 6 **MF2F**: Multi-function interrupt 2 request flag
0: No request
1: Interrupt request

Bit 5 Unimplemented, read as "0"

Bit 4 **MF0F**: Multi-function interrupt 0 request flag
0: No request
1: Interrupt request

Bit 3 **SPIE**: SPI interrupt control
0: Disable
1: Enable

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

Bit 1 Unimplemented, read as "0"

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

• **INTC1 Register – BH66F71662**

Bit	7	6	5	4	3	2	1	0
Name	SPIF	MF2F	MF1F	MF0F	SPIE	MF2E	MF1E	MF0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **SPIF**: SPI interrupt request flag
0: No request
1: Interrupt request

Bit 6 **MF2F**: Multi-function interrupt 2 request flag
0: No request
1: Interrupt request

Bit 5 **MF1F**: Multi-function interrupt 1 request flag
0: No request
1: Interrupt request

Bit 4 **MF0F**: Multi-function interrupt 0 request flag
0: No request
1: Interrupt request

Bit 3 **SPIE**: SPI interrupt control
0: Disable
1: Enable

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

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

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

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	URF	TB1F	TB0F	—	URE	TB1E	TB0E
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **URF**: UART interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **URE**: UART interrupt control
0: Disable
1: Enable

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

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

• **MFI0 Register**

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

Bit 7~6 Unimplemented, read as "0"

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

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

Bit 3~2 Unimplemented, read as "0"

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

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

• **MFI1 Register – BH66F71662**

Bit	7	6	5	4	3	2	1	0
Name	—	—	STMAF	STMPF	—	—	STMAE	STMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"
0: No request
1: Interrupt request

Bit 5 **STMAF**: STM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **STMPF**: STM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **STMAE**: STM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **STMPE**: STM Comparator P match interrupt control
0: Disable
1: Enable

• **MFI2 Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **DEF**: Data EEPROM interrupt request flag
0: No request
1: Interrupt request

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

Bit 3~2 Unimplemented, read as "0"

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

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

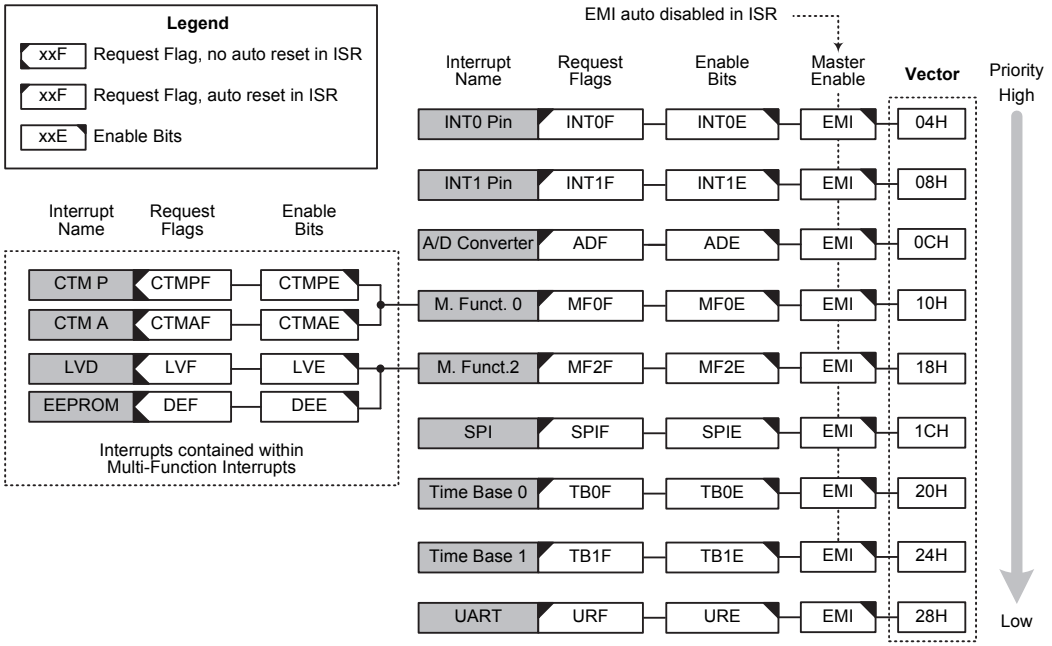
Interrupt Operation

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

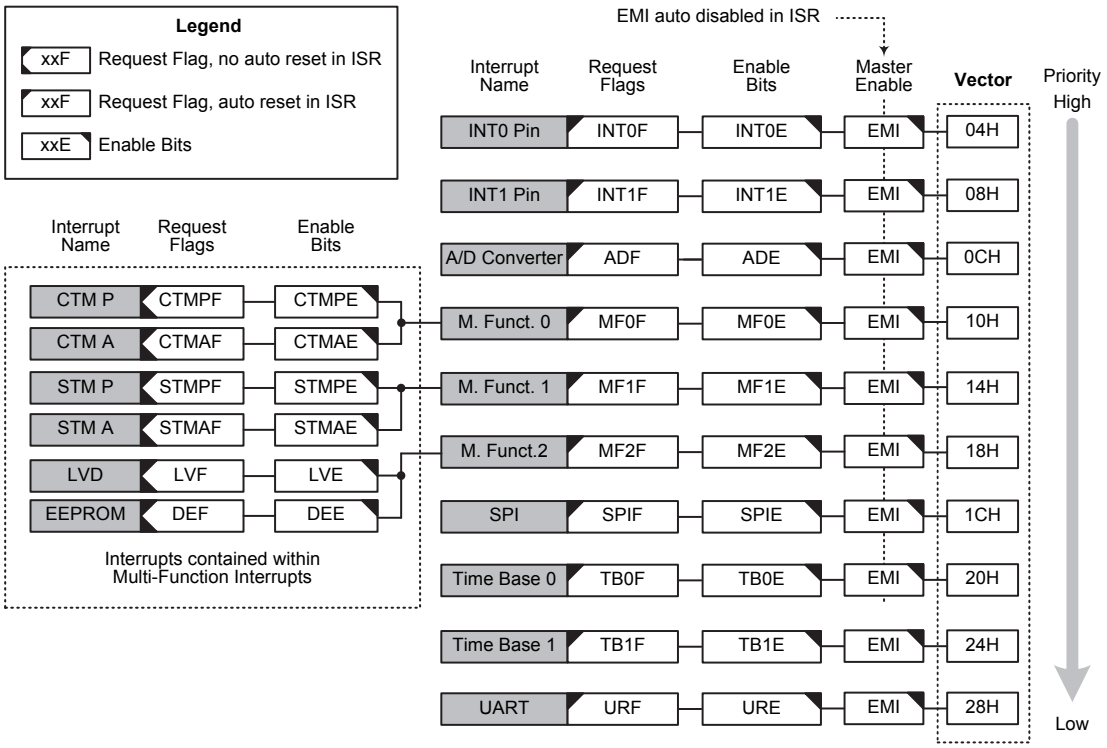
When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will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

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

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.



Interrupt Structure – BH66F71652



Interrupt Structure – BH66F71662

External Interrupts

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

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

A/D Converter Interrupt

An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Interrupt vector, will take place. When the A/D Converter Interrupt is serviced, the A/D Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Serial Peripheral Interface Interrupt

The Serial Peripheral Interface Interrupt, known as the SPI interrupt, will take place when the SPI Interrupt request flag, SPIF, is set, which occurs when a byte of data has been received or transmitted by the SPI interface or an SPI incomplete transfer occurs. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, SPIE, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the respective Interrupt vector, will take place. When the interrupt is serviced, the Serial Interface Interrupt flag, SPIF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

UART Interrupt

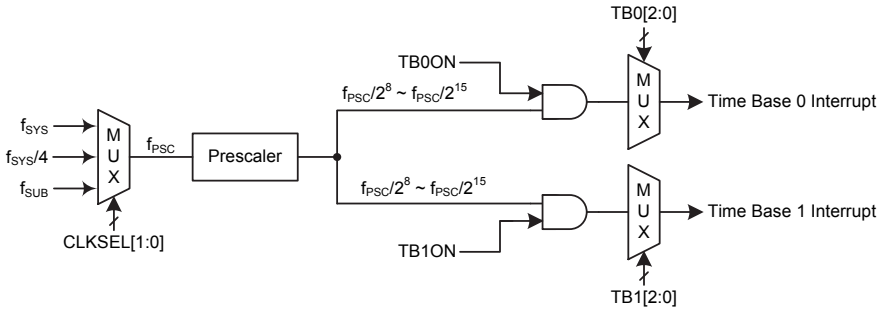
Several individual UART conditions can generate a UART Interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. To allow the program to branch to the respective interrupt vector addresses, the global interrupt enable bit, EMI, and UART Interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of these conditions are created, a subroutine call to the UART Interrupt vector will take place. When the interrupt is serviced, the

UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART section.

Time Base Interrupts

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

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, f_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL1~CLKSEL0 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 selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

• **TB0C Register**

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

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

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Select Time Base 0 time-out period
000: $2^8/f_{PSC}$
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**: Select Time Base 1 time-out period
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: $2^{15}/f_{PSC}$

Multi-function Interrupts

Within the devices there are up to three Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupt, EEPROM Interrupt and LVD Interrupt.

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

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

EEPROM Interrupt

The EEPROM interrupt is contained within the Multi-function Interrupt. An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit, 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 respective Multi-function Interrupt vector will take place. When the EEPROM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

LVD Interrupt

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

TM Interrupts

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

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector 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.

Interrupt Wake-up Function

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

interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

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

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

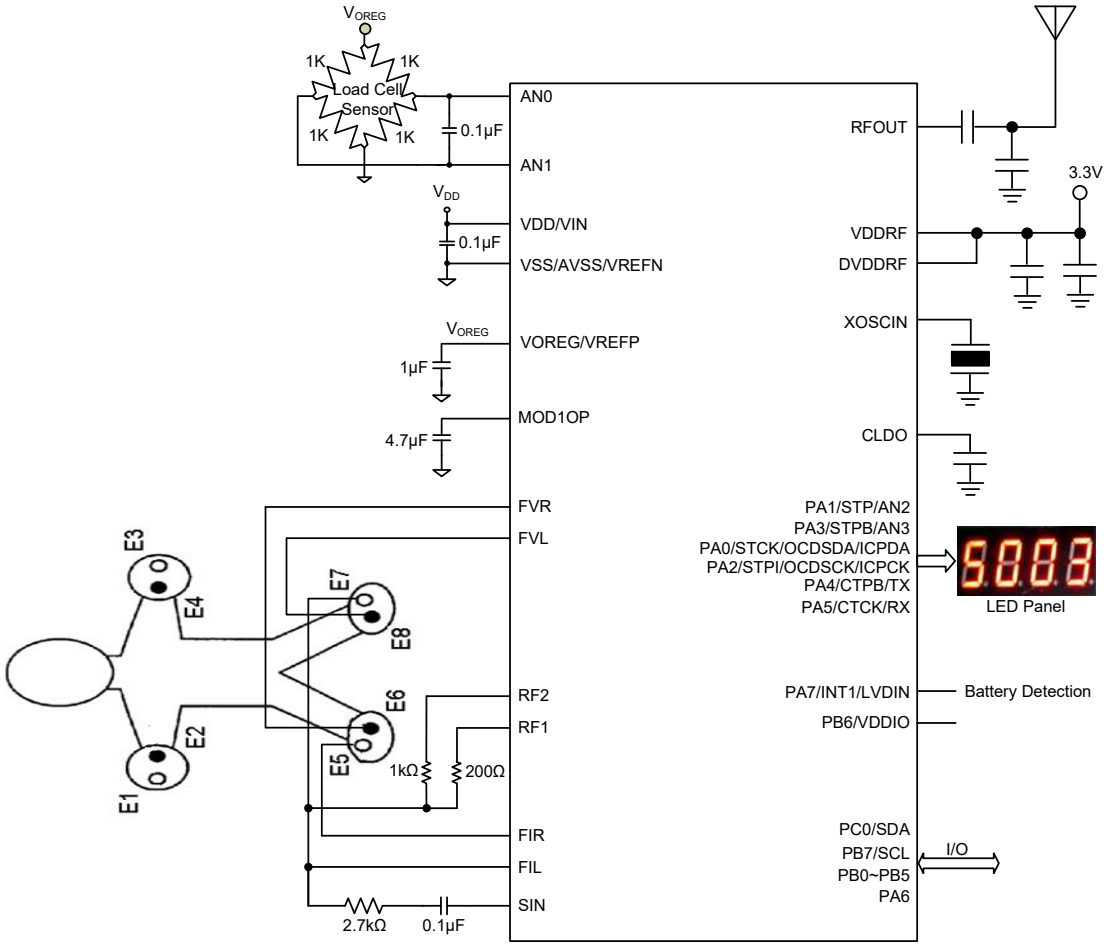
It is recommended that programs do not use the "CALL" instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged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

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

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

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

Application Circuits



Note: The STM related pins are only available in the BH66F71662 device.

Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of several kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

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

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

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

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

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

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	2Note	Z, C, AC, OV, SC
LADC A,[m]	Add Data Memory to ACC with Carry	2	Z, C, AC, OV, SC
LADCM A,[m]	Add ACC to Data memory with Carry	2Note	Z, C, AC, OV, SC
LSUB A,[m]	Subtract Data Memory from ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LSBC A,[m]	Subtract Data Memory from ACC with Carry	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2Note	C
Logic Operation			
LAND A,[m]	Logical AND Data Memory to ACC	2	Z
LOR A,[m]	Logical OR Data Memory to ACC	2	Z
LXOR A,[m]	Logical XOR Data Memory to ACC	2	Z
LANDM A,[m]	Logical AND ACC to Data Memory	2Note	Z
LORM A,[m]	Logical OR ACC to Data Memory	2Note	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2Note	Z
LCPL [m]	Complement Data Memory	2Note	Z
LCPLA [m]	Complement Data Memory with result in ACC	2	Z
Increment & Decrement			
LINCA [m]	Increment Data Memory with result in ACC	2	Z
LINC [m]	Increment Data Memory	2Note	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2Note	Z
Rotate			
LRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2Note	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2Note	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2Note	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2Note	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2Note	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2Note	None
LSET [m].i	Set bit of Data Memory	2Note	None

Mnemonic	Description	Cycles	Flag Affected
Branch			
LSZ [m]	Skip if Data Memory is zero	2Note	None
LSZA [m]	Skip if Data Memory is zero with data movement to ACC	2Note	None
LSNZ [m]	Skip if Data Memory is not zero	2Note	None
LSZ [m].i	Skip if bit i of Data Memory is zero	2Note	None
LSNZ [m].i	Skip if bit i of Data Memory is not zero	2Note	None
LSIZ [m]	Skip if increment Data Memory is zero	2Note	None
LSDZ [m]	Skip if decrement Data Memory is zero	2Note	None
LSIZA [m]	Skip if increment Data Memory is zero with result in ACC	2Note	None
LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC	2Note	None
Table Read			
LTABRD [m]	Read table to TBLH and Data Memory	3Note	None
LTABRDL [m]	Read table (last page) to TBLH and Data Memory	3Note	None
LITABRD [m]	Increment table pointer TBLP first and Read table to TBLH and Data Memory	3Note	None
LITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	3Note	None
Miscellaneous			
LCLR [m]	Clear Data Memory	2Note	None
LSET [m]	Set Data Memory	2Note	None
LSWAP [m]	Swap nibbles of Data Memory	2Note	None
LSWAPA [m]	Swap nibbles of Data Memory with result in ACC	2	None

Note: 1. For these extended skip instructions, if the result of the comparison involves a skip then up to four cycles are required, if no skip takes place two cycles is required.

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

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

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

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

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

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

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

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

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

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

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − $\overline{C}$

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

SBC A, x Subtract immediate data from ACC with Carry

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

Operation ACC ← ACC – [m] – $\overline{C}$

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

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

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

Operation [m] ← ACC − [m] − $\overline{C}$

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

SDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

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

Operation Skip if [m]≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

Description The low byte of the program code (specific page) addressed by the table pointer pair (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

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

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

Affected flag(s) None

ITABRD [m] Increment table pointer low byte first and read table to TBLH and Data Memory

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Extended Instruction Definition

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

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

LADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

LADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″AND″ [m]

Affected flag(s) Z

LCLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

LCLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

LCPL [m] Complement Data Memory

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

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

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

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

Affected flag(s) C

LDEC [m] Decrement Data Memory

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

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

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

LINC [m] Increment Data Memory

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

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

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] + 1$

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

LMOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

LRL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

LRLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

LRR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

LRRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − $\overline{C}$

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

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

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

Operation [m] ← ACC − [m] − $\overline{C}$

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

LSDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

LSET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

LSET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

LSIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

LSIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

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

Operation Skip if [m] ≠ 0

Affected flag(s) None

LSUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

LSUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

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

Operation [m] ← ACC − [m]

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

LSWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

LSZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

Description The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

LTABRDL [m] Read table (last page) to TBLH and Data Memory

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

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

Affected flag(s) None

LITABRD [m] Increment table pointer low byte first and read table to TBLH and Data Memory

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

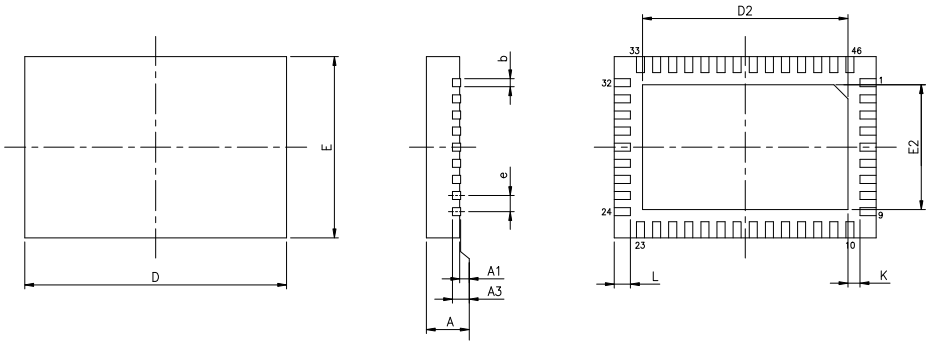
Package Information

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

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

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- Carton information

SAW Type 46-pin QFN (6.5mm×4.5mm×0.75mm) 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 BSC	—
b	0.006	0.008	0.010
D	—	0.256 BSC	—
E	—	0.177 BSC	—
e	—	0.016 BSC	—
D2	0.199	0.201	0.203
E2	0.120	0.122	0.124
L	0.014	0.016	0.018
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 BSC	—
b	0.15	0.20	0.25
D	—	6.50 BSC	—
E	—	4.50 BSC	—
e	—	0.40 BSC	—
D2	5.05	5.10	5.15
E2	3.05	3.10	3.15
L	0.35	0.40	0.45
K	0.20	—	—