HOLTEK

24-Bit Delta Sigma A/D Flash MCU
Integrated Regulator & OPA & LCD Driver

BH67F5250/BH67F5260/BH67F5270

Revision: V1.40 Date: December 02, 2019

www.holtek.com

Table of Contents

Features **7**
CPU Features 7
Peripheral Features 7
Applications **8**
General Description **8**
Block Diagram **9**
Selection Table **10**
Pin Assignment **10**
Pin Description **12**
Absolute Maximum Ratings **17**
D.C. Characteristics **17**
Operating Voltage Characteristics 17
Operating Current Characteristics 18
Standby Current Characteristics 19
A.C. Characteristics **20**
High Speed Internal Oscillator Frequency Accuracy – HIRC 20
Low Speed Internal Oscillator Frequency Accuracy – LIRC 20
Operating Frequency Characteristic Curves 21
System Start Up Time Characteristics 21
Input/Output Characteristics **22**
Input/Output without Multi-power D.C. Characteristics 22
Input/Output with Multi-power D.C. Characteristics 23
Memory Characteristics **24**
LVR/LVD Electrical Characteristics **24**
24-bit Delta Sigma A/D Converter Electrical Characteristics **25**
Effective Number of Bits (ENOB) 27
LCD Characteristics **27**
Power-on Reset Characteristics **29**
System Architecture **29**
Clocking and Pipelining 29
System Clocking and Pipelining 30
Program Counter 30
Stack 31
Arithmetic and Logic Unit – ALU 32
Flash Program Memory **32**
Structure 32
Special Vectors 33
Look-up Table 33
Table Program Example 34

In Circuit Programming – ICP 35
On-Chip Debug Support – OCDS 36
In Application Programming – IAP 36

Data Memory 54
Structure 54
Data Memory Addressing 55
General Purpose Data Memory 55
Special Purpose Data Memory 55

Special Function Register Description 59
Indirect Addressing Registers – IAR0, IAR1, IAR2 59
Memory Pointers – MP0, MP1L, MP1H, MP2L, MP2H 59
Program Memory Bank Pointer – PBP 60
Accumulator – ACC 61
Program Counter Low Register – PCL 61
Look-up Table Registers – TBLP, TBHP, TBLH 61
Status Register – STATUS 62

EEPROM Data Memory 63
EEPROM Data Memory Structure 63
EEPROM Registers 63
Reading Data from the EEPROM 66
Writing Data to the EEPROM 66
Write Protection 66
EEPROM Interrupt 66
Programming Considerations 67

Oscillators 68
Oscillator Overview 68
System Clock Configurations 68
External Crystal/Ceramic Oscillator – HXT 69
Internal High Speed RC Oscillator – HIRC 70
External 32.768kHz Crystal Oscillator – LXT 70
Internal 32kHz Oscillator – LIRC 72

Operating Modes and System Clocks 72
System Clocks 72
System Operation Modes 73
Control Registers 74
Operating Mode Switching 78
Standby Current Considerations 82
Wake-up 82

Watchdog Timer 83
Watchdog Timer Clock Source 83
Watchdog Timer Control Register 83
Watchdog Timer Operation 84

Reset and Initialisation 85
Reset Functions 85
Reset Initial Conditions 88
Input/Output Ports 93
Pull-high Resistors 93
Port A Wake-up 94
I/O Port Control Registers 94
I/O Port Source Current Selection 95
I/O Port Power Source Control 97
Pin-shared Functions 97
I/O Pin Structures 105
Programming Considerations 106
Timer Modules – TM 106
Introduction 106
TM Operation 107
TM Clock Source 107
TM Interrupts 107
TM External Pins 107
Programming Considerations 109
Standard Type TM – STM 110
Standard TM Operation 110
Standard Type TM Register Description 110
Standard Type TM Operation Modes 114
Periodic Type TM – PTM 124
Periodic TM Operation 124
Periodic Type TM Register Description 124
Periodic Type TM Operating Modes 129
Analog to Digital Converter 138
A/D Converter Overview 138
Internal Power Supply 138
A/D Converter Data Rate Definition 140
A/D Converter Register Description 140
A/D Converter Operation 147
Summary of A/D Conversion Steps 148
Programming Considerations 149
A/D Converter Transfer Function 149
A/D Converted Data 150
A/D Converted Data to Voltage 150
Temperature Sensor 150
A/D Conversion Programming Example 151
16-bit Multiplication Division Unit – MDU 152
MDU Registers 152
MDU Operation 153

LCD Driver 155
LCD Display Memory 155
LCD Clock Source 156
LCD Registers 156
LCD Voltage Source and Biasing 158
LCD Reset Function 161
LCD Driver Output 162
LCD Charge Pump 171
Programming Considerations 171
Universal Serial Interface Module – USIM 172
SPI Interface 172
I^2C Interface 180
UART Interface 190
Serial Peripheral Interface – SPIA 205
SPIA Interface Operation 205
SPIA Registers 206
SPIA Communication 209
SPIA Bus Enable/Disable 211
SPIA Operation Steps 211
Error Detection 212
Low Voltage Detector – LVD 213
LVD Register 213
LVD Operation 214
Interrupts 214
Interrupt Registers 214
Interrupt Operation 219
External Interrupts 220
Universal Serial Interface Module Interrupt 221
SPIA Interrupt 221
Time Base Interrupts 221
A/D Converter Interrupt 223
Multi-function Interrupts 223
EEPROM Interrupt 224
LVD Interrupt 224
TM Interrupts 224
Interrupt Wake-up Function 225
Programming Considerations 225
Configuration Options 226
Application Circuits 226
Instruction Set 227
Introduction 227
Instruction Timing 227
Moving and Transferring Data 227

Arithmetic Operations....227
Logical and Rotate Operation....228
Branches and Control Transfer....228
Bit Operations....228
Table Read Operations....228
Other Operations....228
Instruction Set Summary....229
Table Conventions....229
Extended Instruction Set....231
Instruction Definition....233
Extended Instruction Definition....242
Package Information....249
64-pin LQFP (7mm×7mm) Outline Dimensions....250
80-pin LQFP (10mm×10mm) Outline Dimensions....251

Features

CPU Features

- Operating Voltage
  - f_{SYS}=4MHz: 2.2V~5.5V
  - f_{SYS}=8MHz: 2.2V~5.5V
  - f_{SYS}=12MHz: 2.7V~5.5V
  - f_{SYS}=16MHz: 3.3V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator Types
  - External High Speed Crystal – HXT
  - Internal High Speed RC – HIRC
  - External Low Speed 32.768kHz Crystal – LXT
  - 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
- up to 16-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Program Memory: 8K×16~32K×16
- Data Memory: 512×8~2048×8
- True EEPROM Memory: 128×8~512×8
- In Application Programming function – IAP
- Watchdog Timer function
- Up to 46 bidirectional I/O lines
- 8 differential or 16 single-end channel 24-bit resolution Delta Sigma A/D converter
- 16-bit Multiplication Division Unit
- Two pin-shared external interrupts
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output or single pulse output function
  - 1 Standard type 16-bit Timer Module – STM
  - 3 Periodic type 10-bit Timer Modules – PTM0~PTM2
- Universal Serial Interface Module – USIM for SPI, I^2C or UART communication
- Single Serial SPI Interface – SPIA
- LCD Driver function
  - SEGs×COMs: 28×4, 26×6 or 24×8 for BH67F5250
  - SEGs×COMs: 42×4, 40×6 or 38×8 for BH67F5260/BH67F5270

  - Duty type: 1/4 duty, 1/6 duty or 1/8 duty
  - Bias level: 1/3 bias or 1/4 bias
  - Bias type: R type or C type
  - Waveform type: type A or type B
- Dual Time-Base functions for generation of fixed time interrupt signals
- Low Voltage Reset function
- Low Voltage Detect function
- Package types: 64/80-pin LQFP

Applications

- Electronic Scales
- Blood Pressure Meters
- Blood Glucose Meters
- Pressure Switches
- Other Measuring Products

General Description

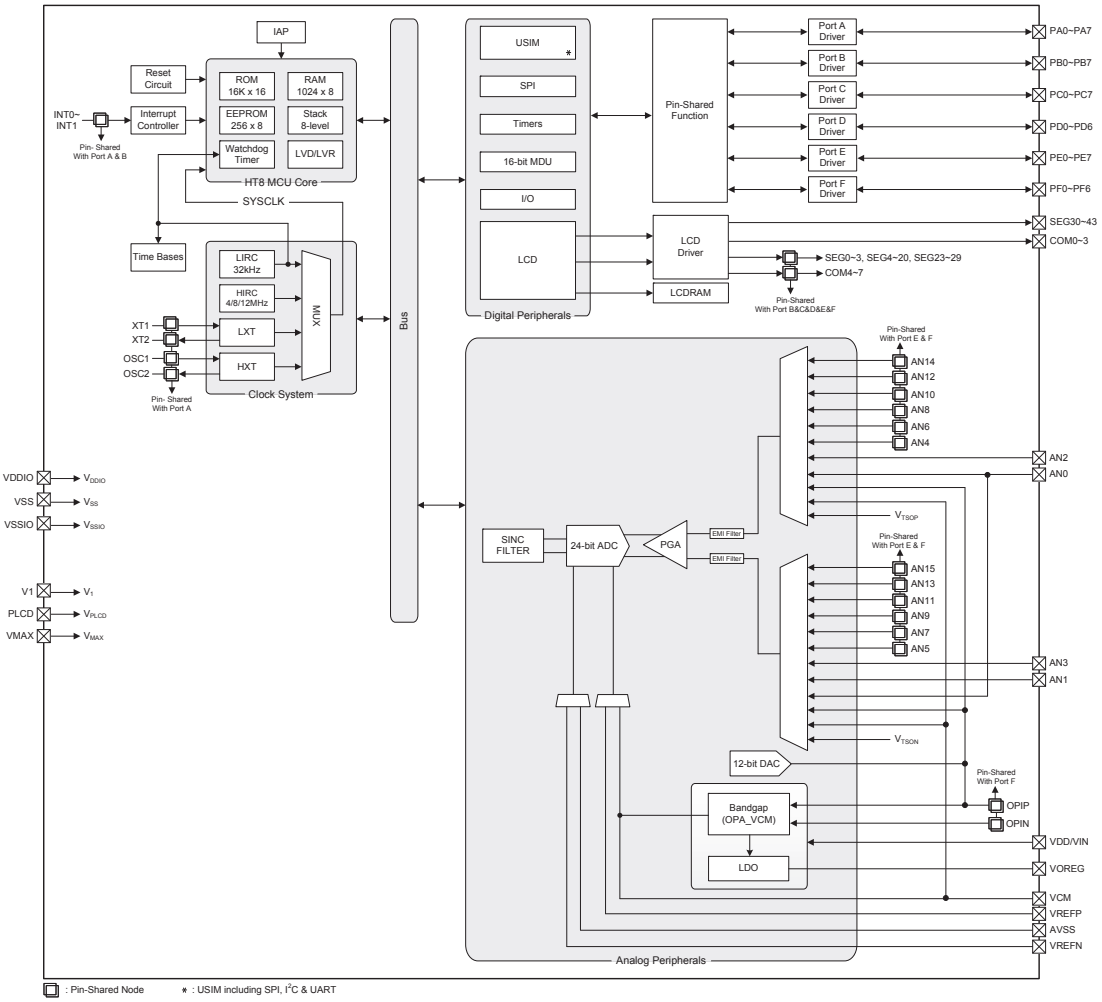
The devices are Flash 8-bit high performance RISC architecture microcontrollers which include a multi-channel 24-bit Delta Sigma A/D converter. This allows them to be used in applications that need to interface to analog signals which require a low noise and high accuracy analog to digital converter. Offering users the convenience of Flash Memory multi-programming features, these devices also include a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel with 24-bit Delta Sigma A/D converter with both single and differential inputs. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI, I^2C and UART interface functions, three popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

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

The inclusion of flexible I/O programming features, a fully integrated LCD driver, a 16-bit MDU and Time-Base funcitons along with many other features enhance the versatility of the devices to enable quick and cost efficient weight measurement scales and other related products.

Block Diagram



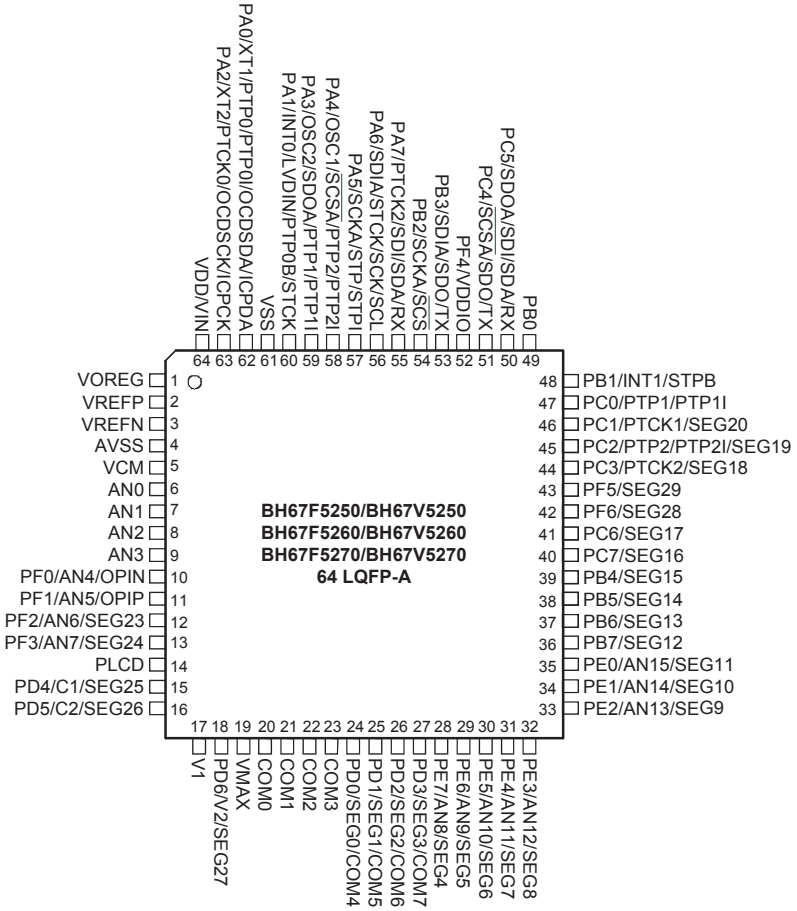
Selection Table

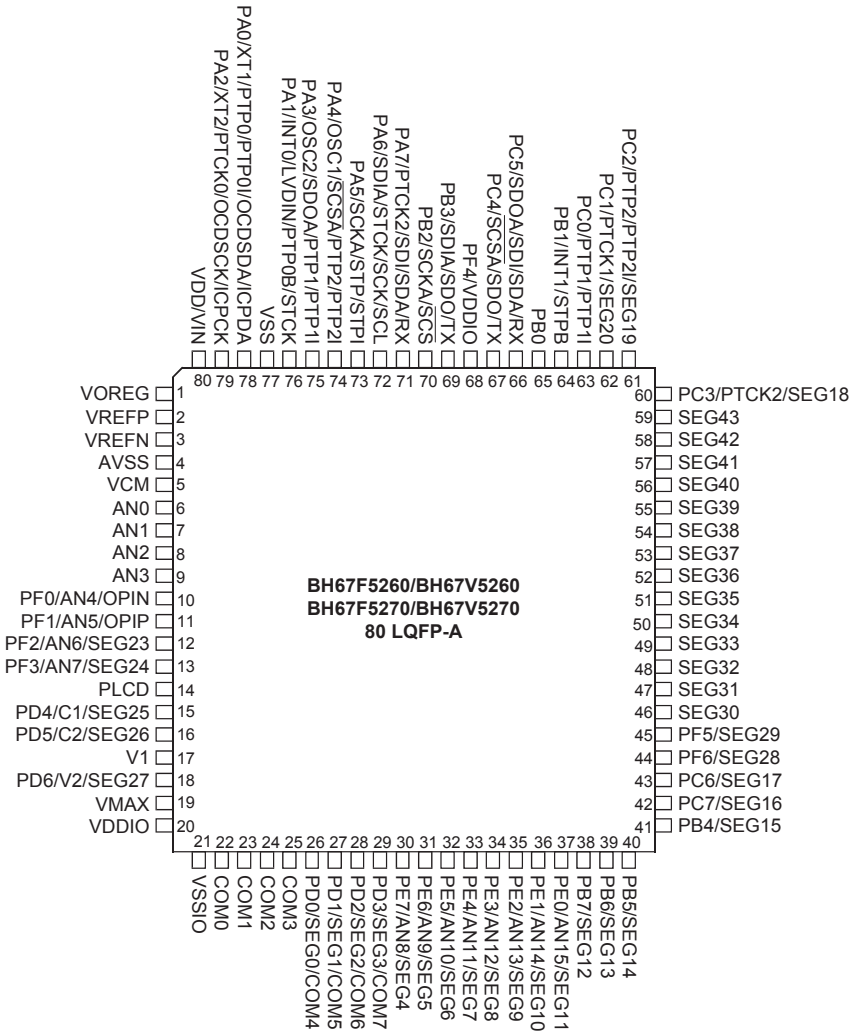
Most features are common to these devices, the main features distinguishing them are Memory capacity, LCD display count, stack capacity and package types. The following table summarises the main features of each device. As devices exist in more than one package format, the tables reflect the situation for the package with the most pins.

Part No.	V_{DD}	Program Memory	Data Memory	Data EEPROM	I/O	IAP	External Interrupt	A/D Converter
BH67F5250	2.2V~5.5V	8K×16	512×8	128×8	46	√	2	24-bit×16
BH67F5260	2.2V~5.5V	16K×16	1024×8	256×8	46	√	2	24-bit×16
BH67F5270	2.2V~5.5V	32K×16	2048×8	512×8	46	√	2	24-bit×16

Part No.	Timer Module	Time Base	USIM	SPI	LCD	16-bit MDU	Stacks	Package
BH67F5250	10-bit PTM×3 16-bit STM×1	2	√	√	28×4/26×6/ 24×8	√	8	64LQFP
BH67F5260	10-bit PTM×3 16-bit STM×1	2	√	√	42×4/40×6/ 38×8	√	8	64/80LQFP
BH67F5270	10-bit PTM×3 16-bit STM×1	2	√	√	42×4/40×6/ 38×8	√	16	64/80LQFP

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 BH67V52x0 devices which are the OCDS EV chips for the BH67F52x0 devices.

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

Pin 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.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/XT1/PTP0/PTP0I/ OCDSDA/ICPDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	XT1	PAS0	LXT	—	LXT oscillator pin
	PTP0	PAS0	—	CMOS	PTM0 output
	PTP0I	PAS0	ST	—	PTM0 capture input
	OCDSDA	—	ST	CMOS	OCDS address/data pin, for EV chip only
	ICPDA	—	ST	CMOS	ICP address/data
PA1/INT0/LVDIN/PTP0B/ STCK	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT0	PAS0 INTEG INTC0	ST	—	External Interrupt 0 input
	LVDIN	PAS0	AN	—	LVD input
	PTP0B	PAS0	—	CMOS	PTM0 inverting output
	STCK	IFS0 PAS0	ST	—	STM clock input
PA2/XT2/PTCK0/OCDSCK/ ICPCK	PA2	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	XT2	PAS0	—	LXT	LXT oscillator pin
	PTCK0	PAS0	ST	—	PTM0 clock input
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only.
	ICPCK	—	ST	—	ICP clock
PA3/OSC2/SDOA/PTP1/ PTP1I	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OSC2	PAS0	—	HXT	HXT pin
	SDOA	PAS0	—	CMOS	SPIA serial data output
	PTP1	PAS0	—	CMOS	PTM1 output
	PTP1I	IFS0 PAS0	ST	—	PTM1 capture input
PA4/OSC1/$\overline{\text{SCSA}}$/PTP2/ PTP2I	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OSC1	PAS1	HXT	—	HXT pin
	$\overline{\text{SCSA}}$	IFS1 PAS1	ST	CMOS	SPIA slave select pin
	PTP2	PAS1	—	CMOS	PTM2 output
	PTP2I	IFS0 PAS1	ST	—	PTM2 capture input

Pin Name	Function	OPT	I/T	O/T	Description
PA5/SCKA/STP/STPI	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCKA	IFS1 PAS1	ST	CMOS	SPIA serial clock input/output
	STP	PAS1	—	CMOS	STM output
	STPI	PAS1	ST	—	STM capture input
PA6/SDIA/STCK/SCK/SCL	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDIA	IFS1 PAS1	ST	—	SPIA serial data input
	STCK	IFS0 PAS1	ST	—	STM clock input
	SCK	PAS1	ST	CMOS	SPI serial clock
	SCL	PAS1	ST	NMOS	I^2C clock line
PA7/PTCK2/SDI/SDA/RX	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PTCK2	IFS0 PAS1	ST	—	PTM2 clock input
	SDI	IFS1 PAS1	ST	—	SPI serial data input
	SDA	IFS1 PAS1	ST	NMOS	I^2C data line
	RX	IFS1 PAS1	ST	—	UART RX serial data input
PB0	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
PB1/INT1/STPB	PB1	PBS0 PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	INT1	PBS0	ST	—	External interrupt 1
	STPB	PBS0	—	CMOS	STM inverting output
PB2/$\overline{SCS}$/SCKA	PB2	PBS0 PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{SCS}$	PBS0	ST	CMOS	SPI slave chip select
	SCKA	IFS1 PBS0	ST	CMOS	SPIA serial clock input/output
PB3/SDIA/SDO/TX	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SDO	PBS0	—	CMOS	SPI serial data output
	SDIA	IFS1 PBS0	ST	—	SPIA serial data input
	TX	PBS0	—	CMOS	UART TX serial data output
PB4/SEG15	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG15	PBS1	—	SEG	LCD Segment output
PB5/SEG14	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG14	PBS1	—	SEG	LCD Segment output
PB6/SEG13	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG13	PBS1	—	SEG	LCD Segment output

Pin Name	Function	OPT	I/T	O/T	Description
PB7/SEG12	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG12	PBS1	—	SEG	LCD Segment output
PC0/PTP1/PTP1I	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1	PCS0	—	CMOS	PTM1 output
	PTP1I	PCS0 IFS0	ST	—	PTM1 capture input
PC1/PTCK1/SEG20	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTCK1	PCS0	ST	—	PTM1 clock input
	SEG20	PCS0	—	SEG	LCD Segment output
PC2/PTP2/PTP2I/SEG19	PC2	PCS0 PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP2	PCS0	—	CMOS	PTM2 output
	PTP2I	IFS0 PCS0	ST	—	PTM2 capture input
	SEG19	PCS0	—	SEG	LCD Segment output
PC3/PTCK2/SEG18	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTCK2	IFS0 PCS0	ST	—	PTM2 clock input
	SEG18	PCS0	—	SEG	LCD Segment output
PC4/$\overline{\text{SCSA}}$/SDO/TX	PC4	PCS1 PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{\text{SCSA}}$	IFS1 PCS1	ST	CMOS	SPIA slave select pin
	SDO	PCS1	—	CMOS	SPI serial data output
	TX	PCS1	—	CMOS	UART TX serial data output
PC5/SDOA/SDI/SDA/RX	PC5	PCS1 PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SDOA	PCS1	—	CMOS	SPIA serial data output
	SDI	IFS1 PCS1	ST	—	SPI serial data input
	SDA	IFS1 PCS1	ST	NMOS	I^2C data line
	RX	IFS1 PCS1	ST	—	UART RX serial data input
PC6/SEG17	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG17	PCS1	—	SEG	LCD Segment output
PC7/SEG16	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG16	PCS1	—	SEG	LCD Segment output
PD0/SEG0/COM4	PD0	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG0	PDS0	—	SEG	LCD Segment output
	COM4	PDS0	—	COM	LCD Common output
PD1/SEG1/COM5	PD1	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG1	PDS0	—	SEG	LCD Segment output
	COM5	PDS0	—	COM	LCD Common output

Pin Name	Function	OPT	I/T	O/T	Description
PD2/SEG2/COM6	PD2	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG2	PDS0	—	SEG	LCD Segment output
	COM6	PDS0	—	COM	LCD Common output
PD3/SEG3/COM7	PD3	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG3	PDS0	—	SEG	LCD Segment output
	COM7	PDS0	—	COM	LCD Common output
PD4/C1/SEG25	PD4	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	C1	PDS1	AN	—	LCD Voltage pump
	SEG25	PDS1	—	SEG	LCD Segment output
PD5/C2/SEG26	PD5	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	C2	PDS1	AN	—	LCD Voltage pump
	SEG26	PDS1	—	SEG	LCD Segment output
PD6/V2/SEG27	PD6	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	V2	PDS1	PWR	AN	LCD voltage pump
	SEG27	PDS1	—	SEG	LCD Segment output
PE0/AN15/SEG11	PE0	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN15	PES0	AN	—	A/D Converter external input
	SEG11	PES0	—	SEG	LCD Segment output
PE1/AN14/SEG10	PE1	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN14	PES0	AN	—	A/D Converter external input
	SEG10	PES0	—	SEG	LCD Segment output
PE2/AN13/SEG9	PE2	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN13	PES0	AN	—	A/D Converter external input
	SEG9	PES0	—	SEG	LCD Segment output
PE3/AN12/SEG8	PE3	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN12	PES0	AN	—	A/D Converter external input
	SEG8	PES0	—	SEG	LCD Segment output
PE4/AN11/SEG7	PE4	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN11	PES1	AN	—	A/D Converter external input
	SEG7	PES1	—	SEG	LCD Segment output
PE5/AN10/SEG6	PE5	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN10	PES1	AN	—	A/D Converter external input
	SEG6	PES1	—	SEG	LCD Segment output
PE6/AN9/SEG5	PE6	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN9	PES1	AN	—	A/D Converter external input
	SEG5	PES1	—	SEG	LCD Segment output

Pin Name	Function	OPT	I/T	O/T	Description
PE7/AN8/SEG4	PE7	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN8	PES1	AN	—	A/D Converter external input
	SEG4	PES1	—	SEG	LCD Segment output
PF0/AN4/OPIN	PF0	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN4	PFS0	AN	—	A/D Converter external input
	OPIN	PFS0	AN	—	OPA negative input
PF1/AN5/OPIP	PF1	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN5	PFS0	AN	—	A/D Converter external input
	OPIP	PFS0	AN	—	OPA positive input
PF2/AN6/SEG23	PF2	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN6	PFS0	AN	—	A/D Converter external input
	SEG23	PFS0	—	SEG	LCD Segment output
PF3/AN7/SEG24	PF3	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN7	PFS0	AN	—	A/D Converter external input
	SEG24	PFS0	—	SEG	LCD Segment output
PF4/VDDIO	PF4	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	VDDIO	PFS1	PWR	—	Positive Power supply
PF5/SEG29	PF5	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG29	PFS1	—	SEG	LCD Segment output
PF6/SEG28	PF6	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG28	PFS1	—	SEG	LCD Segment output
VMAX	VMAX	—	PWR	—	IC maximum voltage, connect to VDD or V1
PLCD	PLCD	—	PWR	AN	LCD power supply
V1	V1	—	PWR	AN	LCD voltage pump
COM0~COM3	COMn	—	—	COM	LCD Common output
SEG30~SEG43 (for BH67F5260/BH67F5270)	SEG30~ SEG43	—	—	SEG	LCD Segment output
VOREG	VOREG	—	—	PWR	LDO output pin
			PWR	—	Positive power supply for VCM, ADC, PGA
AVSS	AVSS	—	PWR	—	Negative power supply for VCM, ADC, PGA
AN0~AN3	AN0~AN3	—	AN	—	A/D Converter external input
VCM	VCM	—	AN	—	External input voltage for ADC Common mode
		—	—	AN	ADC Common mode voltage output
VREFN	VREFN	—	AN	—	ADC external negative reference input
VREFP	VREFP	—	AN	—	ADC external positive reference input
VDD/VIN	VDD	—	PWR	—	Positive Power supply
	VIN	—	PWR	—	LDO input pin
VSS	VSS	—	PWR	—	Negative power supply
VDDIO (for BH67F5260/BH67F5270)	VDDIO	—	PWR	—	Positive Power supply for PA7~PA6, PB3~PB2 and PC5~PC4 pins

Pin Name	Function	OPT	I/T	O/T	Description
VSSIO (for BH67F5260/BH67F5270)	VSSIO	—	PWR	—	Negative Power supply for PA7~PA6, PB3~PB2 and PC5~PC4 pins

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
SEG: LCD SEG output;
COM: LCD COM output;
HXT: High frequency crystal oscillator;
LXT: Low frequency crystal oscillator.

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_{OL} Total 80mA

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

Total Power Dissipation 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the 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 devices 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 – HXT	$f_{SYS}=f_{HXT}$=4MHz	2.2	—	5.5	V
		$f_{SYS}=f_{HXT}$=8MHz	2.2	—	5.5	
		$f_{SYS}=f_{HXT}$=12MHz	2.7	—	5.5	
		$f_{SYS}=f_{HXT}$=16MHz	3.3	—	5.5	
	Operating Voltage – HIRC	$f_{SYS}=f_{HIRC}$=4MHz	2.2	—	5.5	V
		$f_{SYS}=f_{HIRC}$=8MHz	2.2	—	5.5	
		$f_{SYS}=f_{HIRC}$=12MHz	2.7	—	5.5	
	Operating Voltage – LXT	$f_{SYS}=f_{LXT}$=32.768kHz	2.2	—	5.5	V
	Operating Voltage – LIRC	$f_{SYS}=f_{LIRC}$=32kHz	2.2	—	5.5	V

Operating Current Characteristics

Ta=25°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2.2V	f_{SYS}=32kHz	—	8	16	μA
		3V		—	10	20	μA
		5V		—	30	50	μA
	SLOW Mode – LXT	2.2V	f_{SYS}=32.768kHz	—	8	16	μA
		3V		—	10	20	μA
		5V		—	30	50	μA
	FAST Mode – HIRC	2.2V	f_{SYS}=4MHz	—	0.3	0.5	mA
		3V		—	0.4	0.6	mA
		5V		—	0.8	1.2	mA
		2.2V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	mA
		5V		—	1.6	2.4	mA
		2.7V	f_{SYS}=12MHz	—	1.0	1.4	mA
		3V		—	1.2	1.8	mA
		5V		—	2.4	3.6	mA
	FAST Mode – HXT	2.2V	f_{SYS}=4MHz	—	0.4	0.6	mA
		3V		—	0.5	0.75	mA
		5V		—	1	1.5	mA
		2.2V	f_{SYS}=8MHz	—	0.8	1.2	mA
		3V		—	1	1.5	mA
		5V		—	2	3	mA
		2.7V	f_{SYS}=12MHz	—	1.2	2.2	mA
		3V		—	1.5	2.75	mA
		5V		—	3	4.5	mA
		3.3V	f_{SYS}=16MHz	—	3.2	4.8	mA
		5V		—	4	6	mA

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

1. Any digital inputs are setup in a non-floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Operating Current values are measured using a continuous NOP instruction program loop.

Standby Current Characteristics

Ta=25°C

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.2V	WDT off	—	0.11	0.15	2.00	μA
		3V		—	0.11	0.15	2.00	μA
		5V		—	0.18	0.38	2.90	μA
		2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	μA
		5V		—	3	5	6	μA
	IDLE0 Mode – LIRC	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	μA
		5V		—	5	10	12	μA
	IDLE0 Mode – LXT	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	μA
		5V		—	5	10	12	μA
	IDLE1 Mode – HIRC	2.2V	f_{SUB} on, f_{SYS}=4MHz	—	144	200	240	μA
		3V		—	250	360	430	μA
		5V		—	450	720	860	μA
		2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	420	600	720	μA
		5V		—	800	1200	1440	μA
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	600	900	1080	μA
		5V		—	1200	1800	2160	μA
	IDLE1 Mode – HXT	2.2V	f_{SUB} on, f_{SYS}=4MHz	—	144	200	240	μA
		3V		—	250	360	430	μA
		5V		—	450	720	860	μA
		2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	420	600	720	μA
		5V		—	800	1200	1440	μA
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	600	900	1080	μA
		5V		—	1200	1800	2160	μA
		3.3V	f_{SUB} on, f_{SYS}=16MHz	—	1.5	2.0	2.4	mA
		5V		—	2.0	2.8	3.3	mA

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

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

A.C. Characteristics

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

High Speed Internal Oscillator Frequency Accuracy – HIRC

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

4/8/12MHz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	4MHz Writer Trimmed HIRC Frequency	3V/5V	Ta=25˚C	-1%	4	+1%	MHz
			Ta=-40˚C~85˚C	-2%	4	+2%	
		2.2V~5.5V	Ta=25˚C	-2.5%	4	+2.5%	
			Ta=-40˚C~85˚C	-3%	4	+3%	
	8MHz Writer Trimmed HIRC Frequency	3V/5V	Ta=25˚C	-1%	8	+1%	MHz
			Ta=-40˚C~85˚C	-2%	8	+2%	
		2.2V~5.5V	Ta=25˚C	-2.5%	8	+2.5%	
			Ta=-40˚C~85˚C	-3%	8	+3%	
	12MHz Writer Trimmed HIRC Frequency	5V	Ta=25˚C	-1%	12	+1%	MHz
			Ta=-40˚C~85˚C	-2%	12	+2%	
		2.7V~5.5V	Ta=25˚C	-2.5%	12	+2.5%	
			Ta=-40˚C~85˚C	-3%	12	+3%	

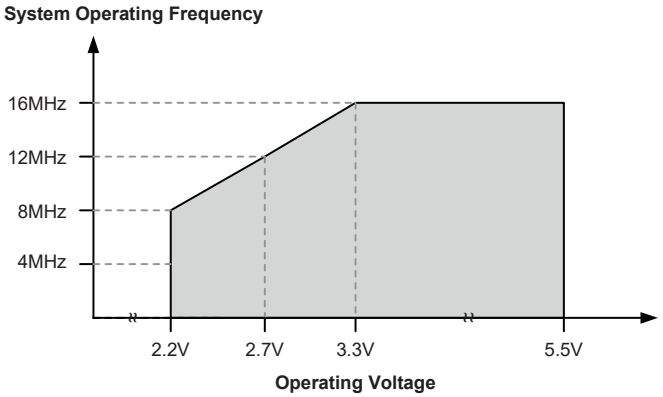
Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.
3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

Low Speed Internal Oscillator Frequency Accuracy – LIRC

Ta=25˚C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	2.2V~5.5V	25˚C	-10%	32	+10%	kHz
			-40˚C~85˚C	-50%	32	+60%	
t_{START}	LIRC Start Up Time	—	—	—	—	500	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	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HXT}$	—	128	—	t_{HXT}
		—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LXT}$	—	1024	—	t_{LXT}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time Wake-up from condition where f_{SYS} is on	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HXT}$ or f_{HIRC}	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LXT}$ or f_{LIRC}	—	2	—	t_{SUB}
	System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode	—	f_{HXT} switches from off → on	—	1024	—	t_{HXT}
		—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
		—	f_{LXT} switches from off → on	—	1024	—	t_{LXT}
t_{RSTD}	System Reset Delay Time Reset Source from Power-on Reset or LVR Hardware Reset	—	RR_{POR}=5V/ms	30	48	72	ms
	System Reset Delay Time LVRC/WDTC/RSTC Software Reset	—	—				
	System Reset Delay Time Reset Source from WDT Overflow	—	—	10	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_{HXT}, 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

except PA7~PA6, PB3~PB2, PC5~PC4 Pins

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	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports or Input Pins	5V	—	3.5	—	5	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Pins	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Pins	3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=00B (n=0, 1, 2; m=0, 2, 4, 6)	-0.7	-1.5	—	mA
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=00B (n=0, 1, 2; m=0, 2, 4, 6)	-1.5	-2.9	—	mA
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=01B (n=0, 1, 2; m=0, 2, 4, 6)	-1.3	-2.5	—	mA
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=01B (n=0, 1, 2; m=0, 2, 4, 6)	-2.5	-5.1	—	mA
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=10B (n=0, 1, 2; m=0, 2, 4, 6)	-1.8	-3.6	—	mA
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=10B (n=0, 1, 2; m=0, 2, 4, 6)	-3.6	-7.3	—	mA
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=11B (n=0, 1, 2; m=0, 2, 4, 6)	-4	-8	—	mA
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=11B (n=0, 1, 2; m=0, 2, 4, 6)	-8	-16	—	mA
R_{PH}	Pull-high Resistance for I/O Ports (Note)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I_{LEAK}	Input Leakage Current	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{INT}	Interrupt Input Pin Minimum Pulse Width	—	—	10	—	—	μs
t_{TPI}	PTPnI and STPI Capture Input Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TCK}	PTCKn and STCK Clock Input Minimum Pulse Width	—	—	0.3	—	—	μs

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

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

PA7~PA6, PB3~PB2, PC5~PC4 Pins

Ta=-40˚C~85˚C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	V_{DD} Power Supply	—	—	2.2	5	5.5	V
V_{DDIO}	V_{DDIO} Power Supply	—	—	2.2	—	V_{DD}	V
V_{IL}	Input Low Voltage for Multi-power I/O Ports	5V	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 Multi-power I/O Ports	5V	Pin power=V_{DD} or V_{DDIO}, V_{DDIO}=V_{DD}	3.5	—	5	V
		—	Pin power=V_{DD} or V_{DDIO}	0.8 (V_{DD}/V_{DDIO})	—	V_{DD}/V_{DDIO}	
I_{OL}	Sink Current for Multi-power I/O Ports	3V	V_{OL}=0.1(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD}	16	32	—	mA
		5V	V_{OL}=0.1(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD}	32	65	—	mA
			V_{OL}=0.1V_{DDIO}, V_{DDIO}=3V	20	40	—	mA
I_{OH}	Source Current for Multi-power I/O Ports	3V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD}, SLEDCn[m+1, m]=00B (n=0, 1, 2; m=0, 2, 4, 6)	-0.7	-1.5	—	mA
		5V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD}, SLEDCn[m+1, m]=00B (n=0, 1, 2; m=0, 2, 4, 6)	-1.5	-2.9	—	mA
			V_{OH}=0.9V_{DDIO}, V_{DDIO}=3V, SLEDCn[m+1, m]=00B (n=0, 1, 2; m=0, 2, 4, 6)	-0.4	-0.85	—	mA
		3V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD}, SLEDCn[m+1, m]=01B (n=0, 1, 2; m=0, 2, 4, 6)	-1.3	-2.5	—	mA
		5V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD}, SLEDCn[m+1, m]=01B (n=0, 1, 2; m=0, 2, 4, 6)	-2.5	-5.1	—	mA
			$V_{OH=}$0.9V_{DDIO}, $V_{DDIO=}$3V, SLEDCn[m+1, m]=01B (n=0, 1, 2; m=0, 2, 4, 6)	-0.7	-1.35	—	mA
		3V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD}, SLEDCn[m+1, m]=10B (n=0, 1, 2; m=0, 2, 4, 6)	-1.8	-3.6	—	mA
		5V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD}, SLEDCn[m+1, m]=10B (n=0, 1, 2; m=0, 2, 4, 6)	-3.6	-7.3	—	mA
			V_{OH}=0.9V_{DDIO}, V_{DDIO}=3V, SLEDCn[m+1, m]=10B (n=0, 1, 2; m=0, 2, 4, 6)	-0.95	-1.9	—	mA
		3V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD}, SLEDCn[m+1, m]=11B (n=0, 1, 2; m=0, 2, 4, 6)	-4	-8	—	mA
		5V	V_{OH}=0.9(V_{DD} or V_{DDIO}), V_{DDIO}=V_{DD}, SLEDCn[m+1, m]=11B (n=0, 1, 2; m=0, 2, 4, 6)	-8	-16	—	mA
			V_{OH}=0.9V_{DDIO}, V_{DDIO}=3V, SLEDCn[m+1, m]=11B (n=0, 1, 2; m=0, 2, 4, 6)	-2.5	-5	—	mA
R_{PH}	Pull-high Resistance for Multi-power I/O Ports	3V	Pin power=V_{DD} or V_{DDIO}, V_{DDIO}=V_{DD}	20	60	100	kΩ
		5V	Pin power=V_{DD} or V_{DDIO}, V_{DDIO}=V_{DD}	10	30	50	kΩ
			Pin power=V_{DDIO}=3V	36	110	180	kΩ

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{LEAK}	Input Leakage Current for Multi-power I/O Ports	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{DDIO}$ or $V_{IN}=V_{SS}$	—	—	±1	μA

Note: The R_{PH} internal pull high resistance value is calculated by connecting to ground and enabled input pin with pull-high resistor and then measuring the input sink 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

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

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	-5%	2.55	+5%	
		—	LVR enable, voltage select 3.15V	-5%	3.15	+5%	
		—	LVR enable, voltage select 3.8V	-5%	3.8	+5%	
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	-5%	2.4	+5%	
		—	LVD enable, voltage select 2.7V	-5%	2.7	+5%	
		—	LVD enable, voltage select 3.0V	-5%	3.0	+5%	
		—	LVD enable, voltage select 3.3V	-5%	3.3	+5%	
		—	LVD enable, voltage select 3.6V	-5%	3.6	+5%	
		—	LVD enable, voltage select 4.0V	-5%	4.0	+5%	
$I_{LVRLVDBG}$	Operating Current	3V	LVD enable, LVR enable, VBGEN=0	—	—	18	μA
		5V	LVD enable, LVR enable, VBGEN=0	—	20	25	μA
		3V	LVD enable, LVR enable, VBGEN=1	—	—	150	μA
		5V	LVD enable, LVR enable, VBGEN=1	—	180	200	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	15	μs
		—	For LVR disable, VBGEN=0, LVD off → on	—	—	150	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs
I_{LVR}	Additional Current for LVR Enable	—	LVD disable, VBGEN=0	—	—	24	μA
I_{LVD}	Additional Current for LVD Enable	—	LVR disable, VBGEN=0	—	—	24	μA
I_{BG}	Additional Current for Bandgap Reference with Buffer	—	LVR disable, LVD disable	—	—	180	μA

24-bit Delta Sigma A/D Converter Electrical Characteristics

$V_{DD}=V_{IN}$, Ta=25°C
LDO & VCM 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	—	5.5	V
I_Q	LDO Quiescent Current	—	LDOVS[1:0]=00B, V_{IN}=3.6V, No load	—	600	720	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	0.21	%/mA
V_{DROP_LDO}	LDO Dropout Voltage(2)	—	LDOVS[1:0]=00B, V_{IN}=3.6V, I_{LOAD}=10mA, △V_{OUT_LDO}=2%	—	—	220	mV
			LDOVS[1:0]=01B, V_{IN}=3.6V, I_{LOAD}=10mA, △V_{OUT_LDO}=2%	—	—	200	
			LDOVS[1:0]=10B, V_{IN}=3.6V, I_{LOAD}=10mA, △V_{OUT_LDO}=2%	—	—	180	
			LDOVS[1:0]=11B, V_{IN}=3.6V, I_{LOAD}=10mA, △V_{OUT_LDO}=2%	—	—	160	
TC_{LDO}	LDO Temperature Coefficient	—	Ta=-40°C~85°C, LDOVS[1:0]=00B, V_{IN}=3.6V, I_{LOAD}=100μA	—	—	0.48	mV/°C
ΔV_{LINE_LDO}	LDO Line Regulation	—	LDOVS[1:0]=00B, 2.6V≤V_{IN}≤5.5V, I_{LOAD}=100μA	—	—	0.7	%/V
		—	LDOVS[1:0]=00B, 2.6V≤V_{IN}≤3.6V, I_{LOAD}=100μA	—	—	0.2	%/V
V_{OUT_VCM}	VCM Output Voltage	—	V_{IN}=3.6V, No load	-5%	1.25	+5%	V
TC_{VCM}	VCM Temperature Coefficient	—	Ta=-40°C~85°C, V_{IN}=3.6V, No load	—	—	200	ppm/°C
ΔV_{LINE_VCM}	VCM Line Regulation	—	2.6V≤V_{IN}≤3.6V, No load	—	—	0.3	%/V
t_{VCMS}	VCM Turn On Stable Time	—	V_{IN}=3.6V, No load	—	—	10	ms

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH}	Source Current for VCM Output Pin	—	V_{IN}=3.6V, $\triangle V_{OUT_VCM}$=-2%	2	—	—	mA
I_{OL}	Sink Current for VCM Output Pin	—	V_{IN}=3.6V, $\triangle V_{OUT_VCM}$=+2%	2	—	—	mA
ADC & ADC Internal Reference Voltage (Delta Sigma A/D Converter)							
V_{OREG}	Supply Voltage for ADC, PGA	—	LDOEN=0	2.4	—	3.3	V
			LDOEN=1	2.4	—	3.3	
I_{ADC}	Additional Current for ADC Enable	—	VRBUFP=1,VRBUFN=1	—	550	700	μA
		—	VRBUFP=0, VRBUFN=0	—	400	550	
I_{ADSTB}	Standby Current	—	MCU enters SLEEP Mode, No load	—	—	1	μA
N_R	Resolution	—	—	—	—	24	bit
INL	Integral Non-linearity	—	V_{OREG}=3.3V, V_{REF}=1.25V, $\triangle$SI=±450mV, PGA gain=1	—	±50	±200	ppm
NFB	Noise Free Bits	—	PGA gain=128 Data rate=10Hz	—	15.4	—	bit
ENOB	Effective Number of Bits	—	PGA gain=128 Data rate=10Hz	—	18.1	—	bit
f_{ADCK}	A/D Converter Clock Frequency	—	—	40	409.6	440	kHz
f_{ADO}	A/D Converter 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}	Reference Input Voltage	—	VRBUFP=0, VRBUFN=0	V_{REFN} +0.8	—	V_{OREG}	V
V_{REFN}		—		0	—	V_{REFP} -0.8	V
V_{REF}		—	V_{REF}=(V_{REFP}-V_{REFN})×VREFGN	0.80	—	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	—	175	—	μV/˚C
OPA							
I_{OPA}	Additional Current for OPA Enable	—	No load	—	200	320	μA
V_{OS}	Input Offset Voltage	—	—	-15	—	+15	mV
$V_{CM\ OPA}$	Common Mode Voltage Range	—	—	V_{SS} +0.3	—	V_{OREG} -1.4	V
PSRR	Power Supply Rejection Ratio	—	—	50	80	—	dB
CMRR	Common Mode Rejection Ratio	—	—	50	80	—	dB
DAC							
V_{DACO}	Output Voltage Range	—	—	V_{SS}	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	V_{OREG}	—	V_{DD}	V
I_{DAC}	Additional Current for DAC Enable	—	V_{REF}=5V	—	—	610	μA
DNL	Differential Nonlinearity	—	2.4V≤V_{DD}≤5.5V	-6	—	+6	LSB
INL	Integral Nonlinearity	—	2.4V≤V_{DD}≤5.5V	-12	—	+12	LSB

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)/θ_{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_{OREG}=2.4V, V_{REF}=1.2V, f_{ADCK}=133kHz

Data Rate (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
4	19.7	19.8	19.6	19.7	19.7	19.6	19.2	18.6
8	19.4	19.3	19.3	19.3	19.3	19.1	18.7	18.1
16	19.0	18.8	18.7	18.9	18.8	18.6	18.2	17.5
33	18.4	18.3	18.3	18.3	18.3	18.1	17.7	17.0
65	18.1	17.9	18.0	17.9	17.9	17.6	17.2	16.5
130	17.6	17.4	17.4	17.4	17.3	17.1	16.6	15.9
260	15.8	15.8	15.9	15.8	15.9	15.9	15.8	15.3
521	14.1	14.0	14.0	14.1	14.1	14.0	14.1	14.4

V_{OREG}=2.4V, V_{REF}=1.2V, f_{ADCK}=333kHz

Data Rate (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
10	19.4	18.8	18.7	18.8	18.8	18.7	18.9	18.1
20	19.0	18.3	18.3	18.3	18.3	18.2	17.9	17.3
41	18.5	17.8	17.8	17.8	17.9	17.7	17.4	16.8
81	18.2	18.2	18.1	18.2	18.1	17.8	17.2	16.4
163	17.9	17.8	17.8	17.8	17.6	17.3	16.7	15.9
326	17.4	17.2	17.2	17.2	17.1	16.8	16.2	15.4
651	16.2	16.1	16.1	16.1	16.1	15.9	15.5	14.8
1302	14.5	14.5	14.5	14.4	14.5	14.5	14.3	14.0

LCD Characteristics

Ta=-25˚C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IN}	LCD Operating Voltage	–	Power supply from PLCD pin (for R type) (Note)	3.0	–	5.5	V
		–	Power supply from PLCD pin (for C type) (Note)	2.0	–	3.7	V
		–	Power supply from V1 pin (for C type) (Note)	3.0	–	5.5	V
		–	Power supply from V2 pin (for C type) (Note)	1.0	–	1.8	V
		–	Power supply from V_A (for C type) (Note)	3.0	–	5.5	V
		3.3V~5.5V	Power supply from V_B (for C type)	-10%	3.0	+10%	V
		2.2V~5.5V	Power supply from V_C (for C type)	-10%	1.08	+10%	V

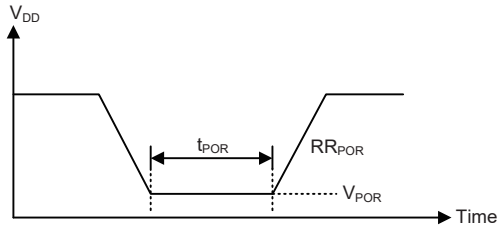
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{LCD}	Additional Current for LCD Enable (R type) LCD Clock=4kHz	5V	No load, $V_A=V_{PLCD}=V_{DD}$, 1/3 Bias, RCT=0, LCDPR=0, LCDIS[1:0]=00B	—	25	37.5	μA
			No load, $V_A=V_{PLCD}=V_{DD}$, 1/4 Bias, RCT=0, LCDPR=0, LCDIS[1:0]=00B	—	18	28	μA
			No load, $V_A=V_{PLCD}=V_{DD}$, 1/3 Bias, RCT=0, LCDPR=0, LCDIS[1:0]=01B	—	50	75	μA
			No load, $V_A=V_{PLCD}=V_{DD}$, 1/4 Bias, RCT=0, LCDPR=0, LCDIS[1:0]=01B	—	37.5	56	μA
			No load, $V_A=V_{PLCD}=V_{DD}$, 1/3 Bias, RCT=0, LCDPR=0, LCDIS[1:0]=10B	—	100	150	μA
			No load, $V_A=V_{PLCD}=V_{DD}$, 1/4 Bias, RCT=0, LCDPR=0, LCDIS[1:0]=10B	—	75	112.5	μA
			No load, $V_A=V_{PLCD}=V_{DD}$, 1/3 Bias, RCT=0, LCDPR=0, LCDIS[1:0]=11B	—	200	300	μA
			No load, $V_A=V_{PLCD}=V_{DD}$, 1/4 Bias, RCT=0, LCDPR=0, LCDIS[1:0]=11B	—	150	225	μA
	Additional Current for LCD Enable (C type)	3V	No load, $V_A=V_1=V_{DD}$, 1/3 Bias	—	10	15	μA
		5V		—	13.5	20	μA
I_{LCDOL}	LCD Common and Segment Sink Current	3V	$V_{OL}=0.1V_{DD}$	210	420	—	μA
		5V		350	700	—	μA
I_{LCDOH}	LCD Common and Segment Source Current	3V	$V_{OH}=0.9V_{DD}$	-80	-160	—	μA
		5V		-180	-360	—	μA
V_{LCD}	PLCD Comes from Charge Pump	2.2V~5.5V	RCT=0, LCDPR=1, CPVS[1:0]=00B, LCDIS[1:0]=11B	-10%	3.3	+10%	V
			RCT=0, LCDPR=1, CPVS[1:0]=01B, LCDIS[1:0]=11B		3.0		
			RCT=0, LCDPR=1, CPVS[1:0]=10B, LCDIS[1:0]=11B		2.7		
		2.7V~5.5V	RCT=0, LCDPR=1, CPVS[1:0]=11B LCDIS[1:0]=11B	-10%	4.5	+10%	

Note: The LCD maximum operating voltage should be less than V_{DD}+2.0V.

Power-on Reset Characteristics

Ta=25°C

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

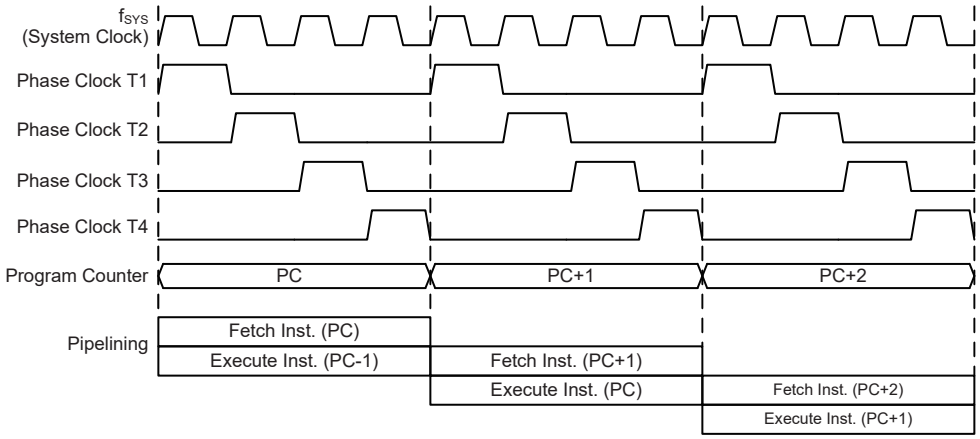


System Architecture

A key factor in the high-performance features of the 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 these devices suitable for low-cost, high-volume production for controller applications.

Clocking and Pipelining

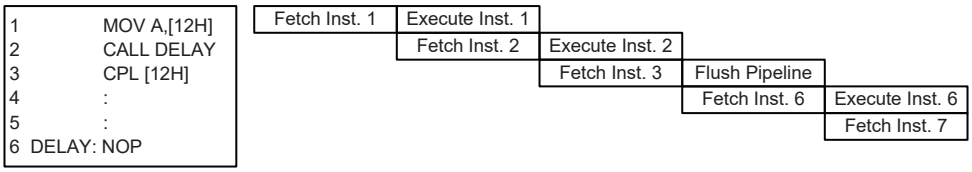
The main system clock, derived from either a HXT, LXT, HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry 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

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 demands a jump to a non-consecutive Program Memory address. As the BH67F5260 and BH67F5270 devices 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 bits, PBP0 and PBP1. 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
BH67F5250	PC12~PC8	PCL7~PCL0
BH67F5260	PBP0, PC12~PC8	PCL7~PCL0

BH67F5270	PBP1, PBP0, PC12~PC8	PCL7~PCL0

Program Counter

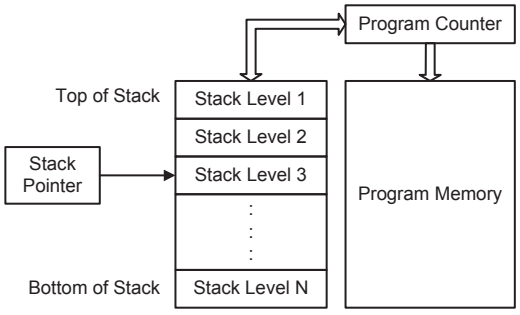
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly. However, as only this low byte is available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

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 up to 16 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, 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.



Note: N=8 for BH67F5250/BH67F5260
N=16 for BH67F5270

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

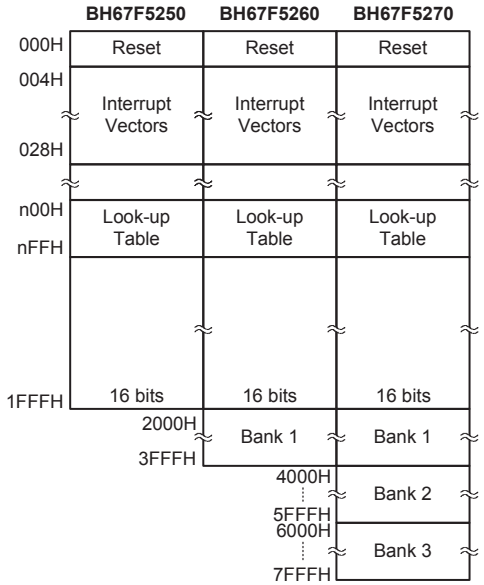
Flash Program Memory

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

Structure

The Program Memory has a capacity of 8K×16 to 32K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.

Device	Capacity	Banks
BH67F5250	8K×16	0
BH67F5260	16K×16	0, 1
BH67F5270	32K×16	0~3



Program Memory Structure

Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by these devices 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.

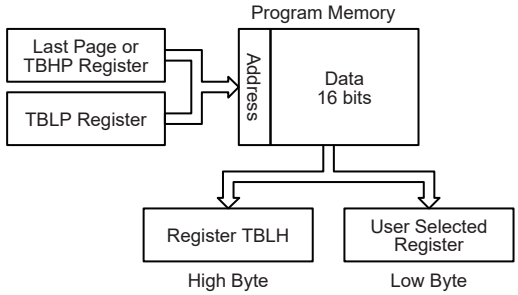


Table Program Example

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 is located in ROM Bank 1 and refers to the start address of the last page within the 16K words Program Memory of the BH67F5260. 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 "3F06H" 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

```
rombank 1 code1
ds .section 'data'
tempreg1 db?       ; temporary register #1
tempreg2 db?       ; temporary register #2
code0 .section 'code'
mov a,06h          ; initialise table pointer - note that this address is referenced
mov tblp,a         ; to the last page or the page that tbhp pointed
mov a,3fh          ; 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 "3F06H" 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 "3F05H" 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

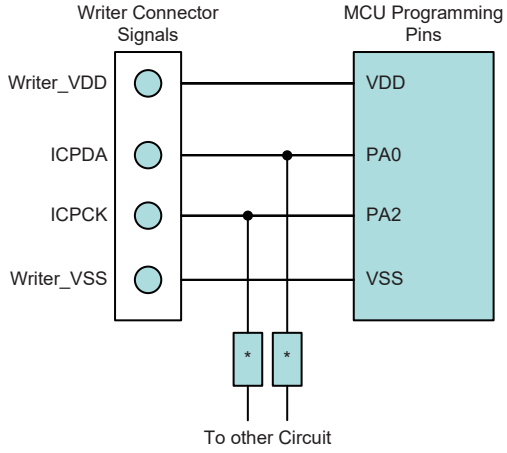
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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 device is beyond the scope of this document and will be supplied in supplementary literature.

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



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

On-Chip Debug Support – OCDS

There is an EV chip named BH67V52x0 which is used to emulate the real MCU device named BH67F52x0. 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 devices, BH67V52x0 and BH67F52x0, 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

In Application Programming – IAP

Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. The provision of the IAP function offers users the convenience of Flash Memory multi-programming features. The convenience of the IAP function is that it can execute the updated program procedure using its internal firmware, without requiring an external Program Writer or PC. In addition, the IAP interface can also be any type of communication protocol, such as UART or USB, using I/O pins. Regarding the internal firmware, the user can select versions provided by HOLTEK or create their own. The following section illustrates the procedures regarding how to implement the IAP firmware.

Flash Memory Read/Write Size

The Flash memory Erase and Write operations are carried out in a page format while the Read operation is carried out in a word format. The page size and write buffer size are both assigned with a capacity of 32 or 64 words respectively. Note that the Erase operation should be executed before the Write operation is executed.

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

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

Device	Program Memory Size	Erase	Write	Read	Page
BH67F5250	8K×16	32 words/Page	32 words/time	1 word/time	32 words
BH67F5260	16K×16	64 words/Page	64 words/time	1 word/time	64 words
BH67F5270	32K×16	64 words/Page	64 words/time	1 word/time	64 words

IAP Operation Format

Erase Page	FARH	FARL[7:5]	FARL[4:0]
0	0000 0000	000	x xxxx
1	0000 0000	001	x xxxx
2	0000 0000	010	x xxxx
3	0000 0000	011	x xxxx
4	0000 0000	100	x xxxx
5	0000 0000	101	x xxxx
6	0000 0000	110	x xxxx
7	0000 0000	111	x xxxx
8	0000 0001	000	x xxxx
9	0000 0001	001	x xxxx
:	:	:	:
126	0000 1111	110	x xxxx
127	0000 1111	111	x xxxx
128	0001 0000	000	x xxxx
129	0001 0000	001	x xxxx
:	:	:	:
254	0001 1111	110	x xxxx
255	0001 1111	111	x xxxx

"x": don't care

Erase Page Number and Selection – BH67F5250

Erase Page	FARH	FARL[7:6]	FARL[5:0]
0	0000 0000	00	xx xxxx
1	0000 0000	01	xx xxxx
2	0000 0000	10	xx xxxx
3	0000 0000	11	xx xxxx
4	0000 0001	00	xx xxxx
5	0000 0001	01	xx xxxx
:	:	:	:
126	0001 1111	10	xx xxxx
127	0001 1111	11	xx xxxx
128	0010 0000	00	xx xxxx
129	0010 0000	01	xx xxxx
:	:	:	:
254	0011 1111	10	xx xxxx
255	0011 1111	11	xx xxxx

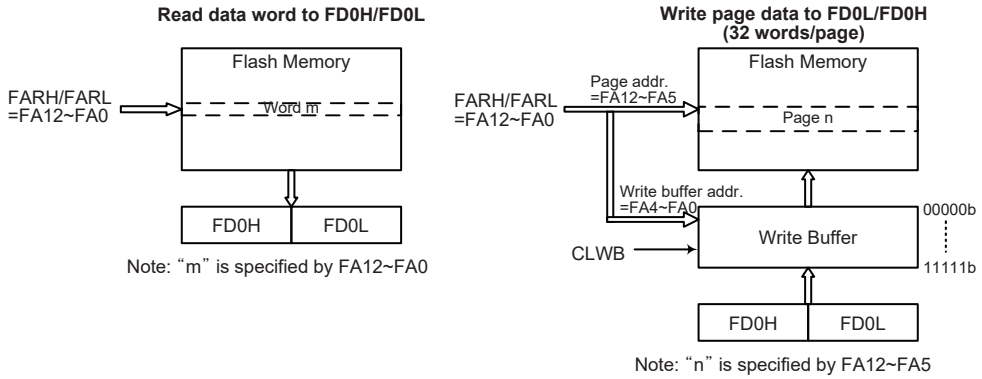
"x": don't care

Erase Page Number and Selection – BH67F5260

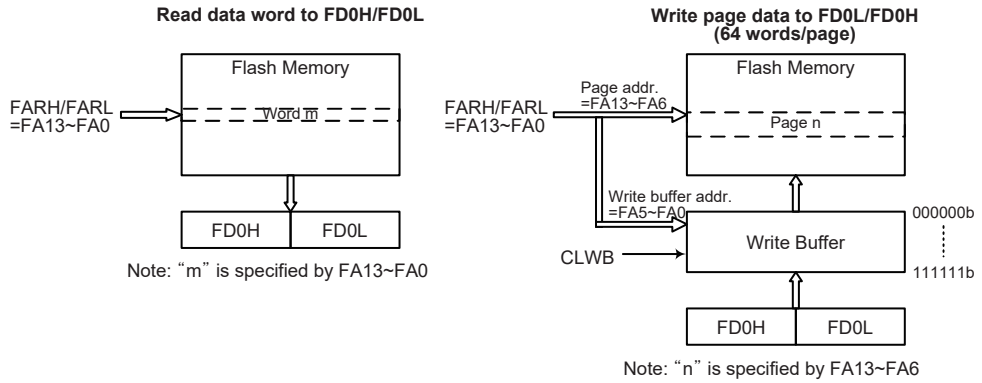
Erase Page	FARH	FARL[7:6]	FARL[5:0]
0	0000 0000	00	xx xxxx
1	0000 0000	01	xx xxxx
2	0000 0000	10	xx xxxx
3	0000 0000	11	xx xxxx
4	0000 0001	00	xx xxxx
5	0000 0001	01	xx xxxx
:	:	:	:
126	0001 1111	10	xx xxxx
127	0001 1111	11	xx xxxx
128	0010 0000	00	xx xxxx
129	0010 0000	01	xx xxxx
:	:	:	:
510	0111 1111	10	xx xxxx
511	0111 1111	11	xx xxxx

"x": don't care

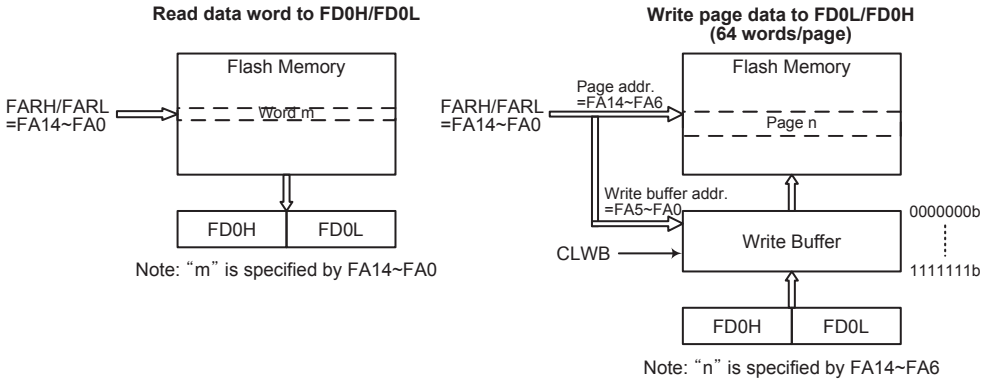
Erase Page Number and Selection – BH67F5270



Flash Memory IAP Read/Write Structure – BH67F5250



Flash Memory IAP Read/Write Structure – BH67F5260



Flash Memory IAP Read/Write Structure – BH67F5270

Write Buffer

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

The write buffer size is 32 or 64 words corresponding to a page respectively. The write buffer address is mapped to a specific flash memory page specified by the memory address bits, FA12~FA5, FA13~FA6 or FA14~FA6. The data written into the FD0L and FD0H registers will be loaded into the write buffer. When data is written into the high byte data register, FD0H, it will result in the data stored in the high and low byte data registers both being written into the write buffer. It will also cause the flash memory address to be incremented by one, after which the new address will be loaded into the FARH and FARL address registers. When the flash memory address reaches the page boundary, 11111b of a page with 32 words or 111111b of a page with 64 words, the address will now not be incremented but will stop at the last address of the page. At this point a new page address should be specified for any other erase/write operations.

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

IAP Flash Program Memory Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	—	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH (BH67F5250)	—	—	—	FA12	FA11	FA10	FA9	FA8
FARH (BH67F5260)	—	—	FA13	FA12	FA11	FA10	FA9	FA8
FARH (BH67F5270)	—	FA14	FA13	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Register List

- **FARL Register**

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

Bit 7~0 Flash Memory Address bit 7 ~ bit 0

- **FARH Register – BH67F5250**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	FA12	FA11	FA10	FA9	FA8
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 Flash Memory Address bit 12 ~ bit 8

- **FARH Register – BH67F5260**

Bit	7	6	5	4	3	2	1	0
Name	—	—	FA13	FA12	FA11	FA10	FA9	FA8
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 Flash Memory Address bit 13 ~ bit 8

• **FARH Register – BH67F5270**

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

Bit 7 Unimplemented, read as "0"

Bit 6~0 Flash Memory Address bit 14 ~ bit 8

• **FD0L Register**

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

Bit 7~0 The first Flash Memory data bit 7 ~ bit 0

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

• **FD0H Register**

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

Bit 7~0 The first Flash Memory data bit 15 ~ bit 8

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

• **FD1L Register**

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

Bit 7~0 The second Flash Memory data bit 7 ~ bit 0

• **FD1H Register**

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

Bit 7~0 The second Flash Memory data bit 15 ~ bit 8

• **FD2L Register**

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

Bit 7~0 The third Flash Memory data bit 7 ~ bit 0

• **FD2H Register**

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

Bit 7~0 The third Flash Memory data bit 15 ~ bit 8

• **FD3L Register**

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

Bit 7~0 The fourth Flash Memory data bit 7 ~ bit 0

• **FD3H Register**

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

Bit 7~0 The fourth Flash Memory data bit 15 ~ bit 8

• **FC0 Register**

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

Bit 7 **CFWEN**: Flash Memory Erase/Write function enable control
0: Flash Memory erase/write function is disabled
1: Flash Memory erase/write function has been successfully enabled
When this bit is cleared to zero by application program, the Flash Memory erase/write function is disabled. Note that this bit cannot be set high by application programs. Writing a "1" into this bit results in no action. This bit is used to indicate that the Flash Memory erase/write function status. When this bit is set high by hardware, it means that the Flash Memory erase/write function is enabled successfully. Otherwise, the Flash Memory erase/write function is disabled as the bit content is zero.

Bit 6~4 **FMOD2~FMOD0**: Flash Memory Mode selection
000: Write Mode
001: Page erase Mode
010: Reserved
011: Read Mode
100: Reserved
101: Reserved
110: Flash Memory Erase/Write function Enable Mode
111: Reserved
These bits are used to select the Flash Memory operation modes. Note that the "Flash memory Erase/Write function Enable Mode" should first be successfully enabled before the Erase or Write Flash memory operation is executed.

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

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

Bit 2 **FWT**: Flash Memory write initiate control
0: Do not initiate Flash Memory write or indicating that a Flash Memory write process has completed
1: Initiate a Flash Memory write process
This bit is set by software and cleared to zero by the hardware when the Flash memory write process has completed. Note that the CPU will be stopped when this bit is set to "1".

Bit 1 **FRDEN**: Flash Memory read enabled bit
0: Flash Memory read disable
1: Flash Memory read enable
This is the Flash memory Read Enable bit which must be set high before any Flash memory read operations are carried out. Clearing this bit to zero will inhibit Flash memory read operations.

Bit 0 **FRD**: Flash Memory read control bit
0: Do not initiate Flash Memory read or indicating that a Flash Memory read process has completed
1: Initiate a Flash Memory read process
This bit is set by software and cleared to zero by the hardware when the Flash memory read process has completed. Note that the CPU will be stopped when this bit is set to "1".

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

• **FC1 Register**

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

Bit 7~0 **D7~D0**: Chip Reset Pattern
When a specific value of "55H" is written into this register, a reset signal will be generated to reset the whole chip.

• **FC2 Register**

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

Bit 7~1 Unimplemented, read as "0"

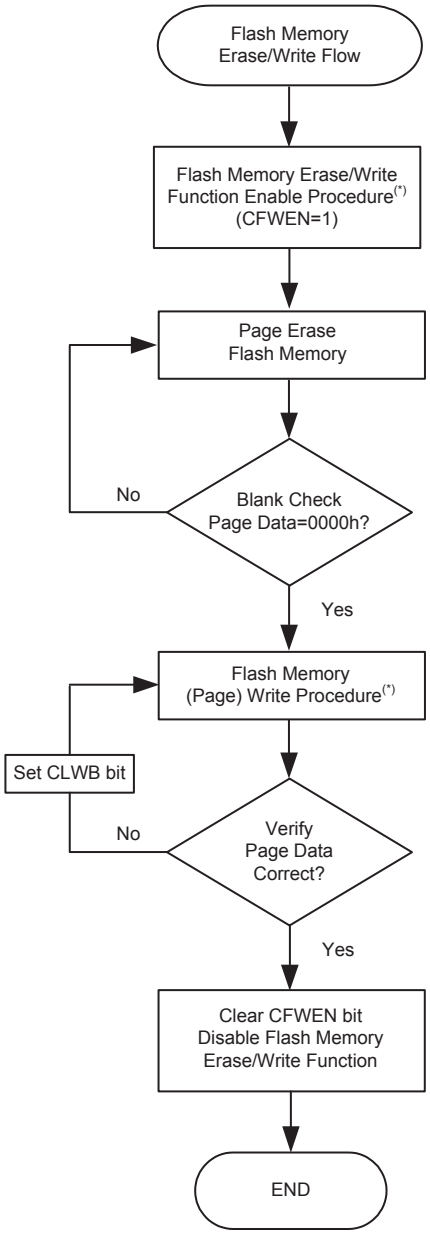
Bit 0 **CLWB**: Flash Memory Write buffer clear control
0: Do not initiate a Write Buffer Clear process or indicating that a Write Buffer Clear process has completed
1: Initiate a Write Buffer Clear process
This bit is set by software and cleared to zero by hardware when the Write Buffer Clear process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

Flash Memory Erase/Write Flow

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

Flash Memory Erase/Write Flow Descriptions

1. Activate the "Flash Memory Erase/Write function enable procedure" first. When the Flash Memory Erase/Write function is successfully enabled, the CFWEN bit in the FC0 register will automatically be set high by hardware. After this, Erase or Write operations can be executed on the Flash memory. Refer to the "Flash Memory Erase/Write Function Enable Procedure" for details.
2. Configure the flash memory address to select the desired erase page and then erase this page.
3. Execute a Blank Check operation to ensure whether the page erase operation is successful or not. The "TABRD" instruction should be executed to read the flash memory contents and to check if the contents is 0000h or not. If the flash memory page erase operation fails, users should go back to Step 2 and execute the page erase operation again.
4. Write data into the specific page. Refer to the "Flash Memory Write Procedure" for details.
5. Execute the "TABRD" instruction to read the flash memory contents and check if the written data is correct or not. If the data read from the flash memory is different from the written data, it means that the page write operation has failed. The CLWB bit should be set high to clear the write buffer and then write the data into the specific page again if the write operation has failed.
6. Clear the CFWEN bit to disable the Flash Memory Erase/Write function enable mode if the current page Erase and Write operations are completed and no more pages need to be erased or written.



Flash Memory Erase/Write Flow

Note: * The Flash Memory Erase/Write Function Enable procedure and Flash Memory Write procedure will be described in the following sections.

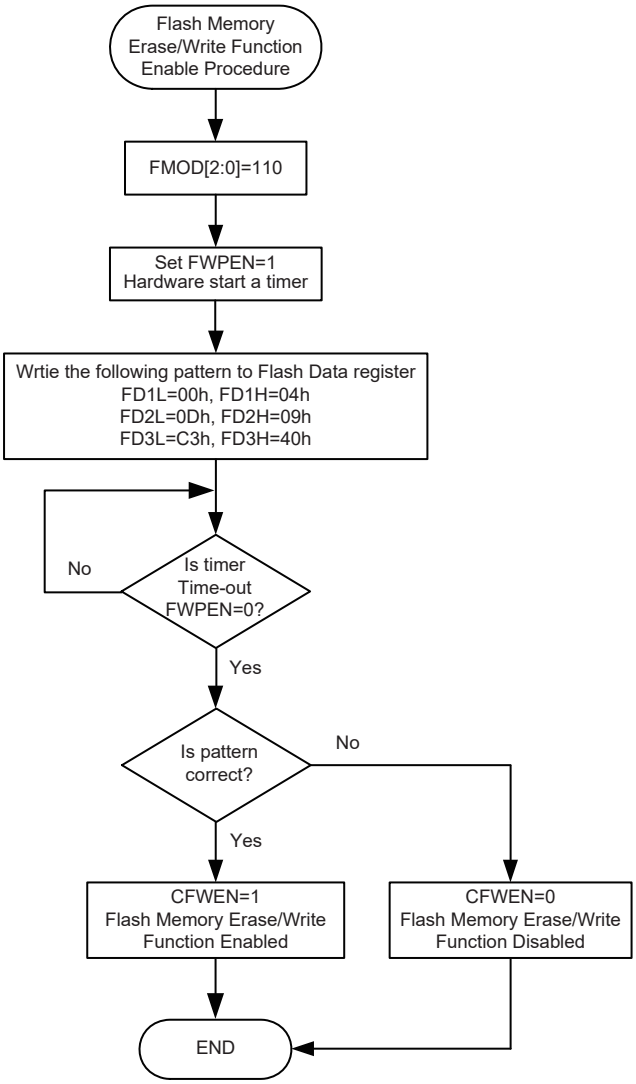
Flash Memory Erase/Write Function Enable Procedure

The Flash Memory Erase/Write Function Enable Mode is specially designed to prevent the flash memory contents from being wrongly modified. In order to allow users to change the Flash memory data using the IAP control registers, users must first enable the Flash memory Erase/Write function.

Flash Memory Erase/Write Function Enable Procedure Description

1. Write data “110” to the FMOD [2:0] bits in the FC0 register to select the Flash Memory Erase/ Write Function Enable Mode.
2. Set the FWPEN bit in the FC0 register to “1” to activate the Flash Memory Erase/Write Enable Function. This will also activate an internal timer.
3. Write the correct data pattern into the Flash data registers, FD1L~FD3L and FD1H~FD3H, as soon as possible after the FWPEN bit is set high. The enable Flash memory erase/write function data pattern is 00H, 0DH, C3H, 04H, 09H and 40H corresponding to the FD1L~FD3L and FD1H~FD3H registers respectively.
4. Once the timer has timed out, the FWPEN bit will automatically be cleared to zero by hardware regardless of the input data pattern.
5. If the written data pattern is incorrect, the Flash memory erase/write function will not be enabled successfully and the above steps should be repeated. If the written data pattern is correct, the Flash memory erase/write function will be enabled successfully.
6. Once the Flash memory erase/write function is enabled, the Flash memory contents can be updated by executing the page erase and write operations using the IAP control registers.

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



Flash Memory Erase/Write Function Enable Procedure

Flash Memory Write Procedure

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

The write buffer size is 32 or 64 words respectively, known as a page, whose address is mapped to a specific flash memory page specified by the memory address bits, FA12~FA5, FA13~FA6 or FA14~FA6. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits, FA12~FA5, FA13~FA6 or FA14~FA6, specify.

Flash Memory Consecutive Write Description

The maximum amount of write data is 32 or 64 words respectively for each write operation. The write buffer address will be automatically incremented by one when consecutive write operations are executed. The start address of a specific page should first be written into the FARL and FARH registers. Then the data word should first be written into the FD0L register and then the FD0H register. At the same time the write buffer address will be incremented by one and then the next data word can be written into the FD0L and FD0H registers for the next address without modifying the address register pair, FARH and FARL. When the write buffer address reaches the page boundary the address will not be further incremented but will stop at the last address of the page.

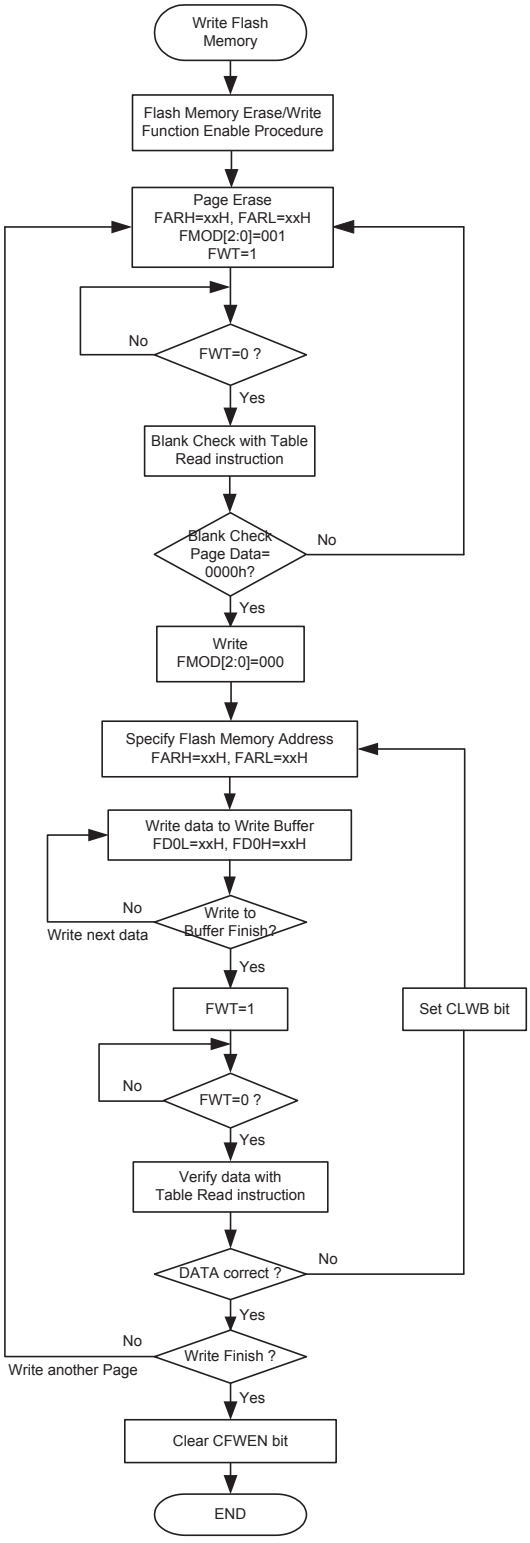
1. Activate the “Flash Memory Erase/Write function enable procedure”. Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the “Flash Memory Erase/Write function enable procedure” for more details.
2. Set the FMOD field to “001” to select the erase operation. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.

   Go to step 2 if the erase operation is not successful.

   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to “000” to select the write operation.
5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L and FD0H registers within a page as specified by their consecutive addresses. The maximum written data number is 32 or 64 words respectively.
6. Set the FWT bit high to write the data words from the write buffer to the flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

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

   Go to step 8 if the write operation is successful.
8. Clear the CFWEN bit low to disable the Flash memory erase/write function.



Flash Memory Consecutive Write Procedure

Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.
2. It will take a typical time of 2.2ms for the FWT bit state changing from high to low.

Flash Memory Non-Consecutive Write Description

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

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

1. Activate the "Flash Memory Erase/Write function enable procedure". Check the CFWEN bit value and then execute the erase/write operation if the CFWEN bit is set high. Refer to the "Flash Memory Erase/Write function enable procedure" for more details.
2. Set the FMOD field to "001" to select the erase operation. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.

   Go to step 2 if the erase operation is not successful.

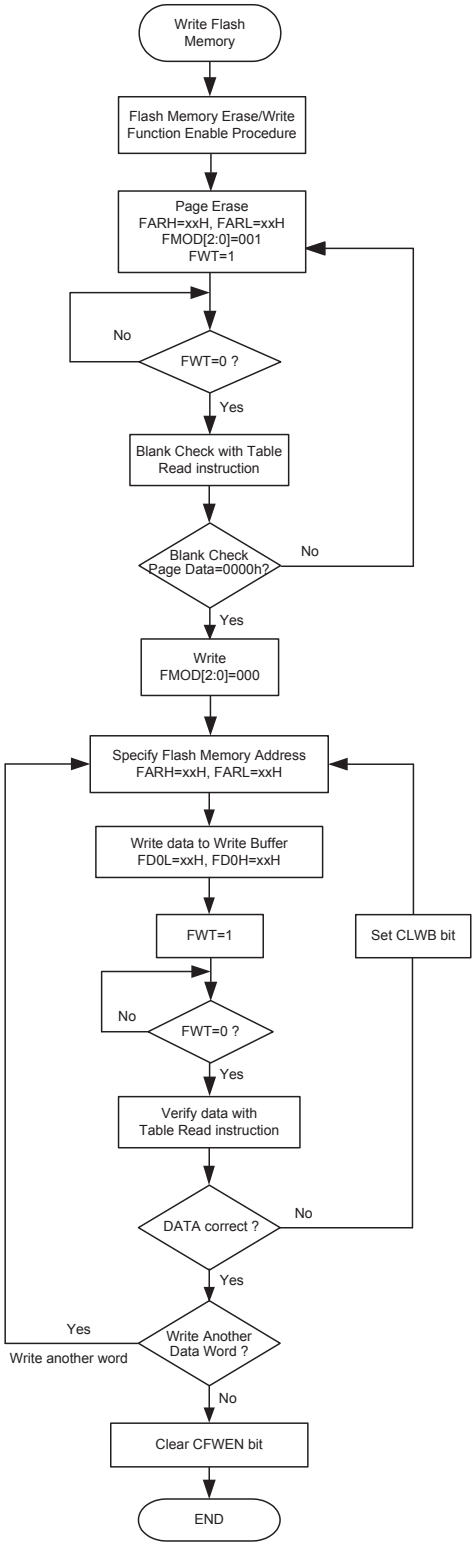
   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to "000" to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
6. Set the FWT bit high to transfer the data word from the write buffer to the flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

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

   Go to step 8 if the write operation is successful.
8. Setup the desired address ADDR2 in the FARH and FARL registers. Write the desired data word DATA2 first into the FD0L register and then into the FD0H register.
9. Set the FWT bit high to transfer the data word from the write buffer to the flash memory. Wait until the FWT bit goes low.
10. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

    If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 8.

    Go to step 11 if the write operation is successful.
11. Clear the CFWEN bit low to disable the Flash memory erase/write function.



Flash Memory Non-Consecutive Write Procedure

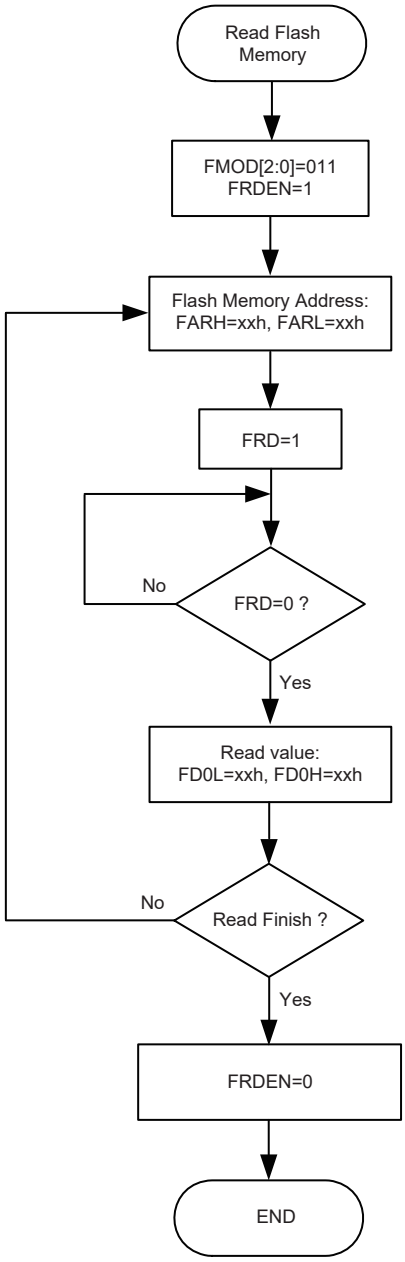
Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.
2. It will take a typical time of 2.2ms for the FWT bit state changing from high to low.

Important Points to Note for Flash Memory Write Operations

1. The "Flash Memory Erase/Write Function Enable Procedure" must be successfully activated before the Flash Memory erase/write operation is executed.
2. The Flash Memory erase operation is executed to erase a whole page.
3. The whole write buffer data will be written into the flash memory in a page format. The corresponding address cannot exceed the page boundary.
4. After the data is written into the flash memory the flash memory contents must be read out using the table read instruction, TABRD, and checked if it is correct or not. If the data written into the flash memory is incorrect, the write buffer should be cleared by setting the CLWB bit high and then write the data again into the write buffer. Then activate a write operation on the same flash memory page without erasing it. The data check, buffer clear and data re-write steps should be repeatedly executed until the data written into the flash memory is correct.
5. The system frequency should be setup to the maximum application frequency when data write and data check operations are executed using the IAP function.

Flash Memory Read Procedure

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



Flash Memory Read Procedure

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

Data Memory

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

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

There is another area of the Data Memory reserved for the LCD display Data Memory. This special area of Data Memory is mapped directly to the LCD display so data written into this memory area will directly affect the displayed data.

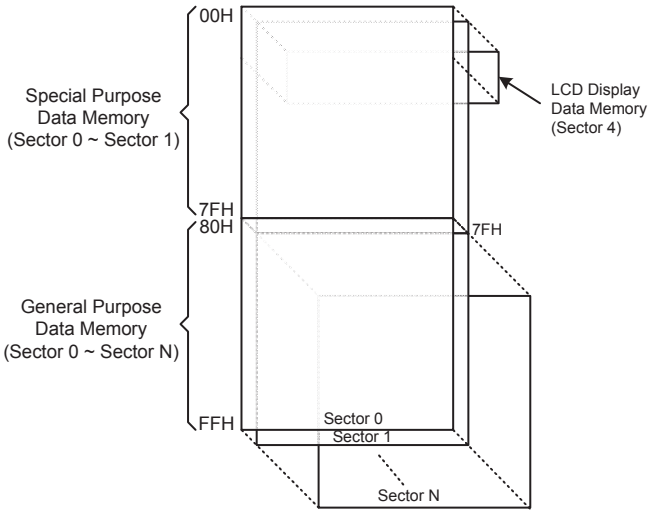
Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value if using the indirect addressing method.

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. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH except the LCD Display Data Memory. The BH67F5250 LCD Display Data Memory is located from 00H to 1FH in Sector 4. For the BH67F5260 and BH67F5270, the LCD Display Data Memory is located from 00H to 2BH in Sector 4.

Device	Special Purpose Data Memory	General Purpose Data Memory		LCD Display Data Memory	
	Located Sectors	Capacity	Sector: Address	Capacity	Sector: Address
BH67F5250	0, 1	512×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH	32×8	4: 00H~1FH
BH67F5260	0, 1	1024×8	0: 80H~FFH 1: 80H~FFH : 6: 80H~FFH 7: 80H~FFH	44×8	4: 00H~2BH
BH67F5270	0, 1	2048×8	0: 80H~FFH 1: 80H~FFH : 14: 80H~FFH 15: 80H~FFH	44×8	4: 00H~2BH

Data Memory Summary



Note: N=3, and Sector 4:00H~1FH for BH67F5250 LCD Display Data Memory;
N=7, and Sector 4:00H~2BH for BH67F5260 LCD Display Data Memory;
N=15, and Sector 4:00H~2BH for BH67F5270 LCD Display Data Memory.

Data Memory Structure

Data Memory Addressing

For these devices that support the extended instructions, there is no Bank Pointer for Data Memory. The Bank Pointer register, PBP, is available for Program Memory for the BH67F5260/BH67F5270. 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 up to 12 valid bits for this series of 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”.

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH		
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	SCC	
11H	HIRCC	
12H	HXTC	
13H	LXTC	
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	RSTC	
19H	LVRC	
1AH	LVDC	
1BH	MFI0	
1CH	MFI1	
1DH	MFI2	
1EH	WDTC	
1FH	INTEG	
20H	INTC0	PTM1C0
21H	INTC1	PTM1C1
22H	INTC2	PTM1DL
23H		PTM1DH
24H	PB	PTM1AL
25H	PBC	PTM1AH
26H	PBPU	PTM1RPL
27H	PC	PTM1RPH
28H	PCC	PTM2C0
29H	PCPU	PTM2C1
2AH		PTM2DL
2BH		PTM2DH
2CH	PSCR	PTM2AL
2DH	TB0C	PTM2AH
2EH	TB1C	PTM2RPL
2FH	SIMC0	PTM2RPH
30H	SIMC1/UUCR1	
31H	SIMA/SIMC2/UUCR2	
32H	SIMD/UTXR_RXR	
33H	SIMTOC/UBRG	
34H	UUSR	
35H		
36H		
37H		
38H		
39H		
3AH	SPIAC0	
3BH	SPIAC1	
3CH	SPIAD	
3DH		
3EH		
3FH		

	Sector 0	Sector 1
40H		EEC
41H	EEA	
42H	EED	
43H		FC0
44H		FC1
45H		FC2
46H		FARL
47H		FARH
48H		FD0L
49H	STMC0	FD0H
4AH	STMC1	FD1L
4BH	STMDL	FD1H
4CH	STMDH	FD2L
4DH	STMAL	FD2H
4EH	STMAH	FD3L
4FH	STMRP	FD3H
50H	PTM0C0	IFS0
51H	PTM0C1	IFS1
52H	PTM0DL	
53H	PTM0DH	PAS0
54H	PTM0AL	PAS1
55H	PTM0AH	PBS0
56H	PTM0RPL	PBS1
57H	PTM0RPH	PCS0
58H		PCS1
59H	MDUWR0	SLEDC0
5AH	MDUWR1	SLEDC1
5BH	MDUWR2	SLEDC2
5CH	MDUWR3	PDS0
5DH	MDUWR4	PDS1
5EH	MDUWR5	PES0
5FH	MDUWCTRL	PES1
60H	PE	PFS0
61H	PEC	PFS1
62H	PEPU	
63H		
64H		
65H	ADCS	
66H	ADCR0	
67H	ADCR1	
68H	PWRC	
69H	PGAC0	
6AH	PGAC1	
6BH	PGACS	
6CH	ADRL	
6DH	ADRM	
6EH	ADRH	
6FH	DSDAH	
70H	DSDAL	
71H	DSDACC	
72H	DSOPC	
73H		
74H		
75H	PD	
76H	PDC	
77H	PDPU	
78H	LCDC0	
79H	LCDCP	
7AH	LCDC2	
7BH	PF	
7CH	PFC	
7DH	PFPU	
7EH	PMPS	
7FH		

: Unused, read as 00H

: Reserved, cannot be changed

Special Purpose Data Memory – BH67F5250

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH	PBP	
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	SCC	
11H	HIRCC	
12H	HXTC	
13H	LXTC	
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	RSTC	
19H	LVRC	
1AH	LVDC	
1BH	MFI0	
1CH	MFI1	
1DH	MFI2	
1EH	WDTC	
1FH	INTEG	
20H	INTC0	PTM1C0
21H	INTC1	PTM1C1
22H	INTC2	PTM1DL
23H		PTM1DH
24H	PB	PTM1AL
25H	PBC	PTM1AH
26H	PBPU	PTM1RPL
27H	PC	PTM1RPH
28H	PCC	PTM2C0
29H	PCPU	PTM2C1
2AH		PTM2DL
2BH		PTM2DH
2CH	PSCR	PTM2AL
2DH	TB0C	PTM2AH
2EH	TB1C	PTM2RPL
2FH	SIMC0	PTM2RPH
30H	SIMC1/UUCR1	
31H	SIMA/SIMC2/UUCR2	
32H	SIMD/UTXR_RXR	
33H	SIMTOC/UBRG	
34H	UUSR	
35H		
36H		
37H		
38H		
39H		
3AH	SPIAC0	
3BH	SPIAC1	
3CH	SPIAD	
3DH		
3EH		
3FH		

	Sector 0	Sector 1
40H		EEC
41H	EEA	
42H	EED	
43H		FC0
44H		FC1
45H		FC2
46H		FARL
47H		FARH
48H		FD0L
49H	STMC0	FD0H
4AH	STMC1	FD1L
4BH	STMDL	FD1H
4CH	STMDH	FD2L
4DH	STMAL	FD2H
4EH	STMAH	FD3L
4FH	STMRP	FD3H
50H	PTM0C0	IFS0
51H	PTM0C1	IFS1
52H	PTM0DL	
53H	PTM0DH	PAS0
54H	PTM0AL	PAS1
55H	PTM0AH	PBS0
56H	PTM0RPL	PBS1
57H	PTM0RPH	PCS0
58H		PCS1
59H	MDUWR0	SLEDC0
5AH	MDUWR1	SLEDC1
5BH	MDUWR2	SLEDC2
5CH	MDUWR3	PDS0
5DH	MDUWR4	PDS1
5EH	MDUWR5	PES0
5FH	MDUWCTRL	PES1
60H	PE	PFS0
61H	PEC	PFS1
62H	PEPU	
63H		
64H		
65H	ADCS	
66H	ADCR0	
67H	ADCR1	
68H	PWRC	
69H	PGAC0	
6AH	PGAC1	
6BH	PGACS	
6CH	ADRL	
6DH	ADRM	
6EH	ADRH	
6FH	DSDAH	
70H	DSDAL	
71H	DSDACC	
72H	DSOPC	
73H		
74H		
75H	PD	
76H	PDC	
77H	PDPU	
78H	LCDC0	
79H	LCDCP	
7AH	LCDC2	
7BH	PF	
7CH	PFC	
7DH	PFPU	
7EH	PMPS	
7FH		

: Unused, read as 00H

: Reserved, cannot be changed

Special Purpose Data Memory – BH67F5260

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH	PBP	
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	SCC	
11H	HIRCC	
12H	HXTC	
13H	LXTC	
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	RSTC	
19H	LVRC	
1AH	LVDC	
1BH	MFI0	
1CH	MFI1	
1DH	MFI2	
1EH	WDTC	
1FH	INTEG	
20H	INTC0	PTM1C0
21H	INTC1	PTM1C1
22H	INTC2	PTM1DL
23H		PTM1DH
24H	PB	PTM1AL
25H	PBC	PTM1AH
26H	PBPU	PTM1RPL
27H	PC	PTM1RPH
28H	PCC	PTM2C0
29H	PCPU	PTM2C1
2AH		PTM2DL
2BH		PTM2DH
2CH	PSCR	PTM2AL
2DH	TB0C	PTM2AH
2EH	TB1C	PTM2RPL
2FH	SIMC0	PTM2RPH
30H	SIMC1/UUCR1	
31H	SIMA/SIMC2/UUCR2	
32H	SIMD/UTXR_RXR	
33H	SIMTOC/UBRG	
34H	UUSR	
35H		
36H		
37H		
38H		
39H		
3AH	SPIAC0	
3BH	SPIAC1	
3CH	SPIAD	
3DH		
3EH		
3FH		

	Sector 0	Sector 1
40H		EEC
41H	EEAL	
42H	EEAH	
43H	EED	FC0
44H		FC1
45H		FC2
46H		FARL
47H		FARH
48H		FD0L
49H	STMC0	FD0H
4AH	STMC1	FD1L
4BH	STMDL	FD1H
4CH	STMDH	FD2L
4DH	STMAL	FD2H
4EH	STMAH	FD3L
4FH	STMRP	FD3H
50H	PTM0C0	IFS0
51H	PTM0C1	IFS1
52H	PTM0DL	
53H	PTM0DH	PAS0
54H	PTM0AL	PAS1
55H	PTM0AH	PBS0
56H	PTM0RPL	PBS1
57H	PTM0RPH	PCS0
58H		PCS1
59H	MDUWR0	SLEDC0
5AH	MDUWR1	SLEDC1
5BH	MDUWR2	SLEDC2
5CH	MDUWR3	PDS0
5DH	MDUWR4	PDS1
5EH	MDUWR5	PES0
5FH	MDUWCTRL	PES1
60H	PE	PFS0
61H	PEC	PFS1
62H	PEPU	
63H		
64H		
65H	ADCS	
66H	ADCR0	
67H	ADCR1	
68H	PWRC	
69H	PGAC0	
6AH	PGAC1	
6BH	PGACS	
6CH	ADRL	
6DH	ADRM	
6EH	ADRH	
6FH	DSDAH	
70H	DSDAL	
71H	DSDACC	
72H	DSOPC	
73H		
74H		
75H	PD	
76H	PDC	
77H	PDPU	
78H	LCDC0	
79H	LCDCP	
7AH	LCDC2	
7BH	PF	
7CH	PFC	
7DH	PFPU	
7EH	PMPS	
7FH		

: Unused, read as 00H

: Reserved, cannot be changed

Special Purpose Data Memory – BH67F5270

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                 ; increment memory pointer
     sdz block               ; check if last memory location has been cleared
     jmp loop
continue:
```

Indirect Addressing Program Example 2

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

The important point to note here is that in the 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 BH67F5260/BH67F5270 devices the Program Memory is divided into several banks. Selecting the required Program Memory area is achieved using the Program Memory Bank Pointer, PBP. The PBP register should be properly configured before the device executes the “Branch” operation using the “JMP” or “CALL” instruction. After that a jump to a non-consecutive Program Memory address which is located in a certain bank selected by the program memory bank pointer bits will occur.

• **PBP Register – BH67F5260**

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

• **PBP Register – BH67F5270**

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

Bit 7~2 Unimplemented, read as "0"

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

Accumulator – ACC

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

Program Counter Low 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

These 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.

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 128×8 bits or 512×8 bits for this series of 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 for BH67F5250 and BH67F5260 or two address registers for BH67F5270 in Sector 0 and a single control register in Sector 1.

Device	Capacity
BH67F5250	128×8
BH67F5260	256×8
BH67F5270	512×8

EEPROM Registers

Three or four registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA for BH67F5250 and BH67F5260 or registers, EEAL and EEAH for BH67F5270, the data register, EED and a single control register, EEC. As the EEA or the EEAL, EEAH and EED registers are located in Sector 0, they can be directly accessed in the same way

as any other Special Function Register. The EEC register however, being located in Sector 1, can only be read from or written to indirectly using the 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(BH67F5250)	—	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EEA(BH67F5260)	EEA7	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EEAL (BH67F5270)	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH (BH67F5270)	—	—	—	—	—	—	—	EEAH0
EED	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM Register List

- **EEA Register – BH67F5250**

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

Bit 7 Unimplemented, read as "0"

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

- **EEA Register – BH67F5260**

Bit	7	6	5	4	3	2	1	0
Name	EEA7	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **EEA7~EEA0**: Data EEPROM address bit 7 ~ bit 0

- **EEAL Register – BH67F5270**

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

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

- **EEAH Register – BH67F5270**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **EEAH0**: Data EEPROM low byte address 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 for BH67F5250 and BH67F5260 or the EEAL and EEAH registers for BH67F5270. 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 for BH67F5250 and BH67F5260 or the EEAL and EEAH registers for BH67F5270 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 exist. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process.

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

Programming Examples – BH67F5250

Reading data from the EEPROM – polling method

```
MOV  A, EEPROM_ADRES            ; user defined address
MOV  EEA, A
MOV  A, 40H                     ; setup memory pointer MP1L
MOV  MP1L, A                    ; MP1L points to EEC register
MOV  A, 01H                     ; setup memory pointer 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 MP1L
MOV  MP1L, A                    ; MP1L points to EEC register
MOV  A, 01H                     ; setup memory pointer 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 a combination of configuration options and relevant control registers.

Oscillator Overview

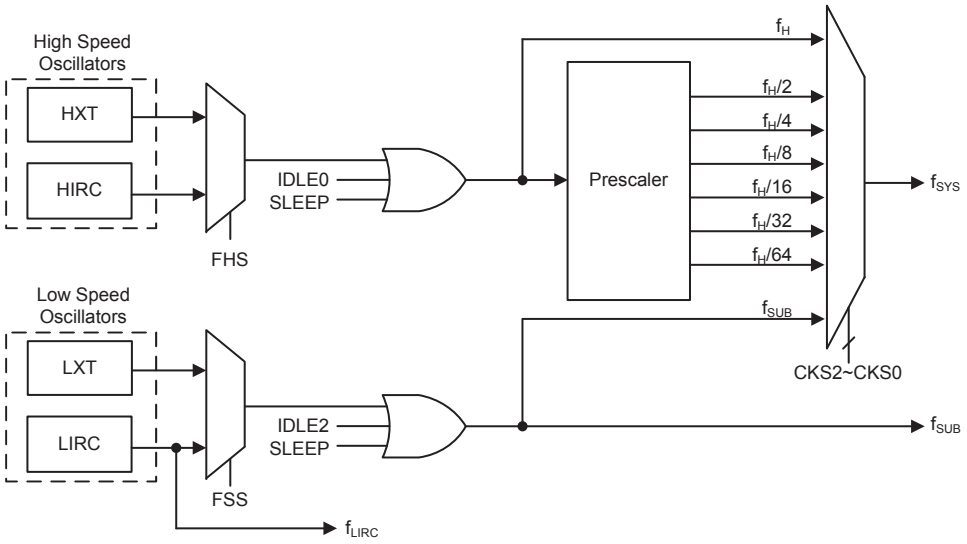
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided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

Type	Name	Frequency	Pins
External High Speed Crystal	HXT	400kHz~16MHz	OSC1/OSC2
Internal High Speed RC	HIRC	4/8/12MHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
Internal Low Speed RC	LIRC	32kHz	—

System Clock Configurations

There are several oscillator sources, two high speed oscillators and two low speed oscillators. The high speed system clocks are sourced from the external crystal/ceramic oscillator, HXT, and the internal 4/8/12MHz RC oscillator, HIRC. The low speed oscillators are the external 32.768kHz crystal oscillator, LXT, and the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

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

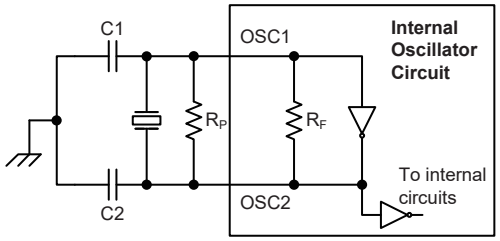


System Clock Configurations

External Crystal/Ceramic Oscillator – HXT

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

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



Note: 1. R_P is normally not required. C1 and C2 are required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

Crystal/Resonator Oscillator – HXT

Crystal Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
16MHz	0pF	0pF
12MHz	0pF	0pF
8MHz	0pF	0pF
6MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 4MHz, 8MHz and 12MHz, which is selected using a configuration option. The HIRC1~HIRC0 bits in the HIRCC register must also be setup to match the selected configuration option frequency. Setting up these bits is necessary to ensure that the HIRC frequency accuracy specified in the A.C. Characterisitcs is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that if this internal system clock option is selected, it requires no external pins for its operation, I/O pins are free for use as normal I/O pins.

External 32.768kHz Crystal Oscillator – LXT

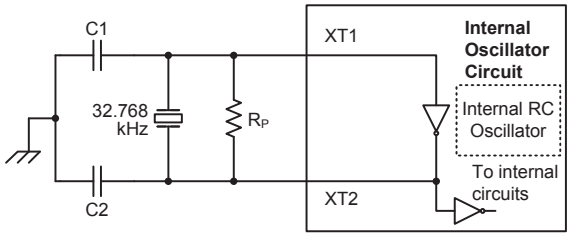
The External 32.768kHz Crystal System Oscillator is one of the low frequency oscillator choices, which is selected via a software control bit, FSS. This clock source has a fixed frequency of 32.768kHz and requires a 32.768kHz crystal to be connected between pins XT1 and XT2. The external resistor and capacitor components connected to the 32.768kHz crystal are necessary to provide oscillation. For applications where precise frequencies are essential, these components may be required to provide frequency compensation due to different crystal manufacturing tolerances. After the LXT oscillator is enabled by setting the LXTEN bit to 1, there is a time delay associated with the LXT oscillator waiting for it to start-up.

However, for some crystals, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. For these devices, the external parallel feedback resistor, R_P is required. For the BH67F5270, the pull high resistor, R_U, are also required except the resistor R_P.

The pin-shared software control bits determine if the XT1/XT2 pins are used for the LXT oscillator or as I/O or other pin-shared functional pins.

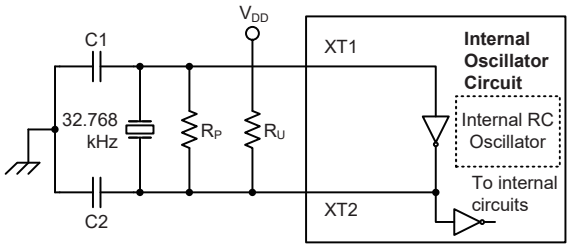
- If the LXT oscillator is not used for any clock source, the XT1/XT2 pins can be used as normal I/O or other pin-shared function pins.
- If the LXT oscillator is used for any clock source, the 32.768kHz crystal should be connected to the XT1/XT2 pins.

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



Note: 1. R_P, C1 and C2 are required.
2. Although not shown XT1/XT2 pins have a parasitic capacitance of around 7pF.

External LXT Oscillator – BH67F5250/BH67F5260



Note: 1. R_P, R_U, C1 and C2 are required.
2. Although not shown XT1/XT2 pins have a parasitic capacitance of around 7pF.

External LXT Oscillator – BH67F5270

LXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
32.768kHz	10pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. R_P=5M~10MΩ is recommended. 2. R_U=10MΩ is recommended only for the BH67F5270.		

32.768kHz Crystal Oscillator Recommended Capacitor Values

LXT Oscillator Low Power Function for BH67F5250/BH67F5260

The LXT oscillator can function in one of two modes, the Speed-Up Mode and the Low-Power Mode. The mode selection is executed using the LXTSP bit in the LXTC register.

LXTSP Bit	LXT Operating Mode
0	Low Power
1	Speed Up

When the LXTSP bit is set to high, the LXT Speed Up Mode will be enabled. In the Speed-Up Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up, it can be placed into the Low-Power Mode by clearing the LXTSP bit to zero and the oscillator will continue to run but with reduced current consumption. It is important to note that the LXT operating mode switching must be properly controlled before the LXT oscillator clock is selected as the system clock source. Once the LXT oscillator clock is selected as the system clock source using the CKS bit field and FSS bit in the SCC register, the LXT oscillator operating mode cannot be changed.

It should be note, that no matter what condition the LXTSP is set to, the LXT oscillator will be always function normally. The only difference is that it will take more time to start up if in the Low Power Mode.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is one of the low frequency oscillator choices, which is selected via a software control bit, FSS. 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.

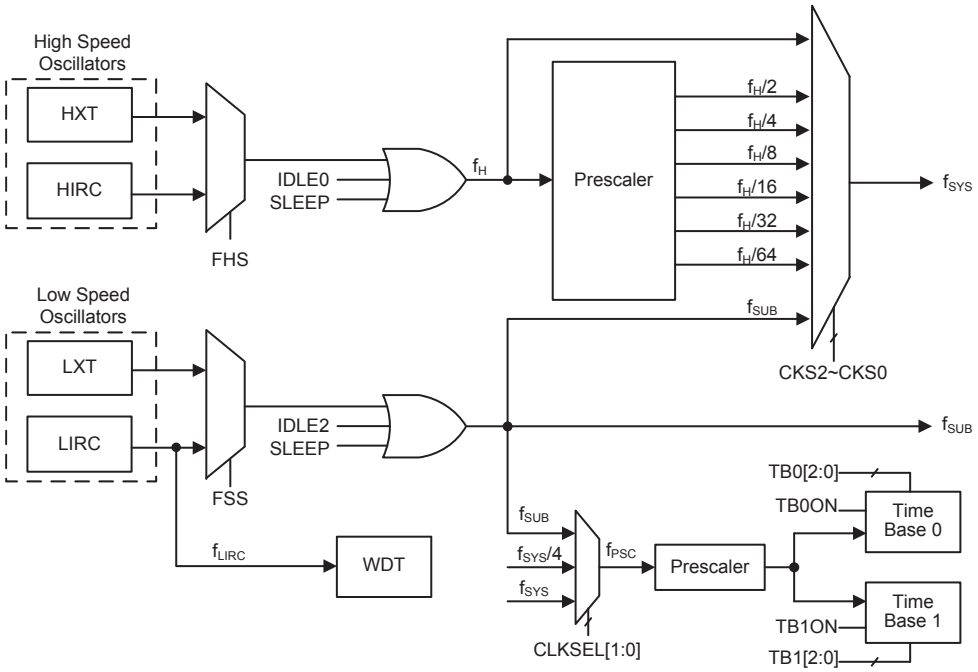
Operating Modes and System Clocks

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 these 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

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

The main system clock, can come from a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock can be sourced from an HXT or HIRC oscillator, selected via configuring the FHS bit in the SCC register. The 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 either the LXT or LIRC oscillators, selected via configuring the FSS bit in the SCC register. 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

As the name suggests 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 one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from one of the high speed oscillators, either the HXT or HIRC oscillators. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from either the LIRC or LXT oscillator determined by the FSS bit in the SCC register.

SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bit are low. In the SLEEP mode the CPU will be stopped. 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, HIRCC, HXTC and LXTC, are used to control the system clock and the corresponding oscillator configurations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	FHS	FSS	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
HXTC	—	—	—	—	—	HXTM	HXTF	HXTEN
LXTC (BH67F5250/5260)	—	—	—	—	—	LXTSP	LXTF	LXTEN
LXTC (BH67F5270)	—	—	—	—	—	—	LXTF	LXTEN

System Operating Mode Control Register List

• **SCC Register**

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	FHS	FSS	FHIDEN	FSIDEN
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 7~5 **CKS2~CKS0**: System clock selection
000: f_H
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: f_{SUB}
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from f_H or f_{SUB}, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as "0"

Bit 3 **FHS**: High Frequency clock selection
0: HIRC
1: HXT

Bit 2 **FSS**: Low Frequency clock selection
0: LIRC
1: LXT

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	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection
00: 4MHz
01: 8MHz
10: 12MHz
11: 4MHz
When the HIRC oscillator is enabled or the HIRC frequency selection is changed by application program, the clock frequency will automatically be changed after the HIRCF flag is set high.
It is recommended that the HIRC frequency selected by these two bits should be the same with the frequency determined by the configuration option to achieve the HIRC frequency accuracy specified in the A.C. Characteristics. Note that these bits are not used to select the oscillator frequency.

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 or the HIRC frequency selection is changed by application program, 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

• **HXTC Register**

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

Bit 7~3 Unimplemented, read as "0"

Bit 2 **HXTM**: HXT mode selection
0: HXT frequency ≤ 10MHz
1: HXT frequency > 10MHz
This bit is used to select the HXT oscillator operating mode. Note that this bit must be properly configured before the HXT is enabled. When the OSC1 and OSC2 pins are enabled and the HXTEN bit is set high to enable the HXT oscillator, it is invalid to change the value of this bit. Otherwise, this bit value can be changed with no operation on the HXT function.

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

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

• **LXTC Register – BH67F5250/BH67F5260**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	LXTSP	LXTF	LXTEN
R/W	—	—	—	—	—	R/W	R	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LXTSP**: LXT speed up control
0: Disable – Low power
1: Enable – Speed up
This bit is used to control whether the LXT oscillator is operating in the low power or Speed-Up mode. When the LXTSP bit is set high, the LXT oscillator will oscillate quickly but consume more power. If the LXTSP bit is cleared to zero, the LXT oscillator will consume less power but take longer time to stablise. It is important to note that this bit cannot be changed after the LXT oscillator is selected as the system clock source using the CKS2~CKS0 and FSS bits in the SCC register.

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

Bit 0 **LXTEN**: LXT oscillator enable control
0: Disable
1: Enable

• **LXTC Register – BH67F5270**

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

Bit 7~2 Unimplemented, read as "0".

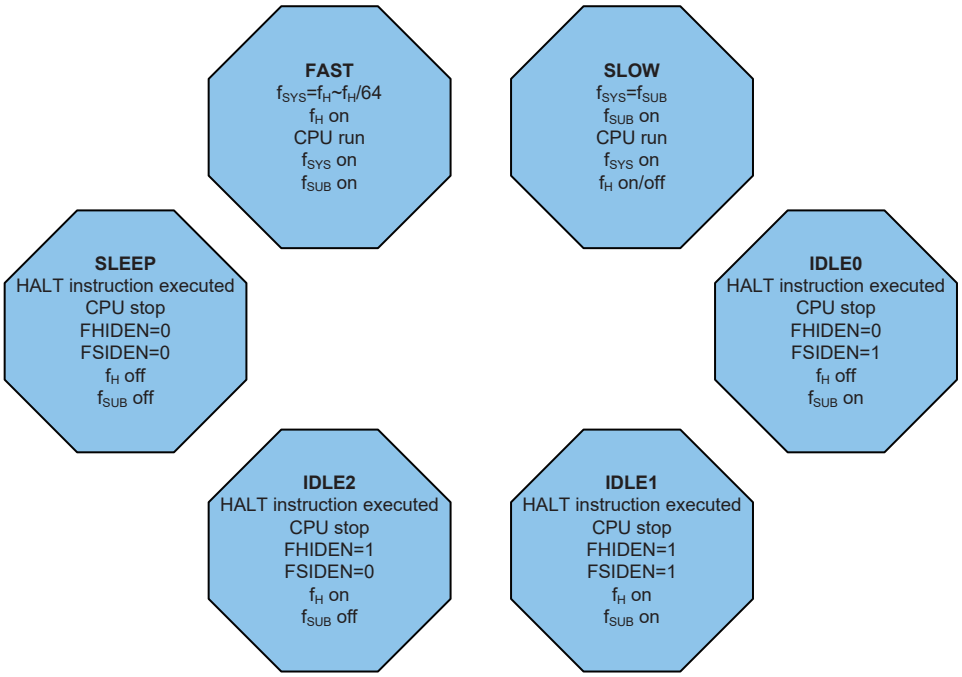
Bit 1 **LXTF**: LXT oscillator stable flag
0: LXT unstable
1: LXT stable
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set high to enable the LXT oscillator, the LXTF bit will first be cleared to 0 and then set high after the LXT oscillator is stable.

Bit 0 **LXTEN**: LXT oscillator enable control
0: Disable
1: Enable

Operating Mode Switching

These 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.

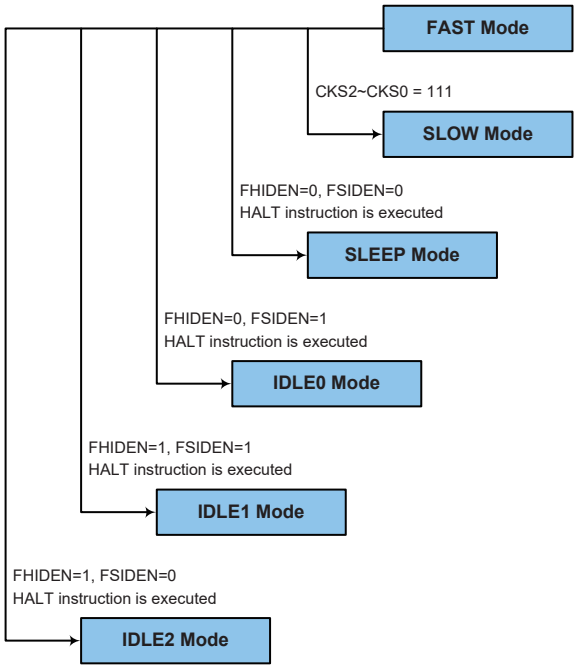
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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

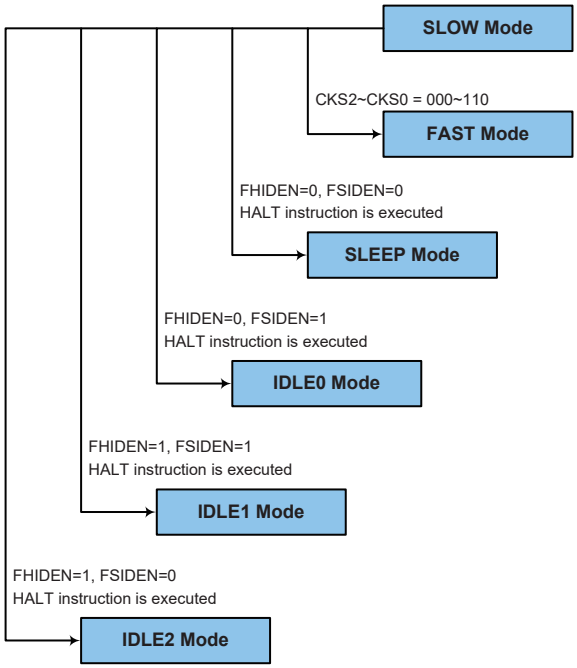
The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires this oscillator to be stable before full mode switching occurs.



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 HXTF bit in the HXTC register or the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the System Start Up Time Characteristics.



Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

- The f_H clock will be stopped and the application program will stop at the "HALT" instruction, but the f_{SUB} clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and 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 device to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

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

Entering the IDLE2 Mode

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

- The f_H clock will be on but the f_{SUB} clock will be off and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and 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 LXT or 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 high. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction. If the system is woken up by a WDT overflow, a Watchdog Time-out hardware 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 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 instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control and reset control of the Watchdog Timer. The WDT function 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.

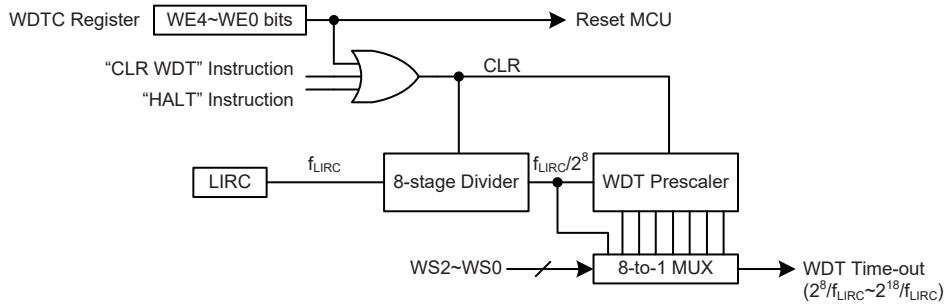
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

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

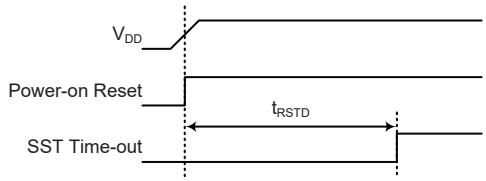
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

Reset Functions

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

Power-on Reset

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



Power-on Reset Timing Chart

Internal Reset Control

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

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

Internal Reset Function Control

• **RSTC Register**

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

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

• **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.

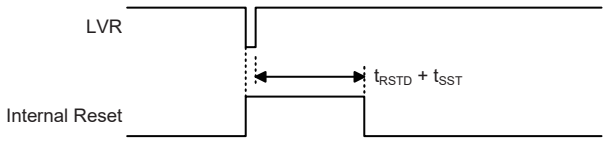
Bit 2 **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.

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

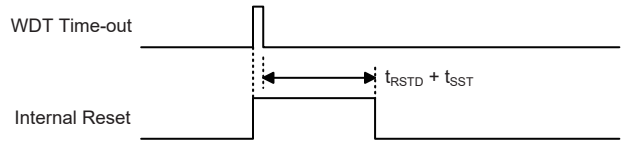
Bit 0 **WRF**: WDT control register software reset flag
Refer to the Watchdog Timer Control Register section.

IAP Reset

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

Watchdog Time-out Reset during Normal Operation

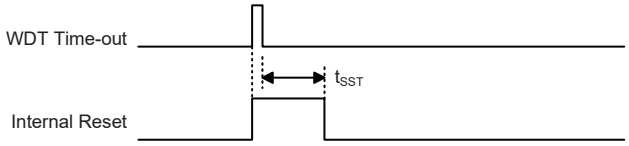
The Watchdog time-out Reset during normal operations in the FAST or SLOW mode is the same as a LVR reset except that the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during 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 Base	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.

Register	BH67F5250	BH67F5260	BH67F5270	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
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
			●	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	●	●	●	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
PBP		●		---- ---0	---- ---0	---- ---0	---- ---u
			●	---- --00	---- --00	---- --00	---- --uu
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- 0000	000- 0000	000- 0000	uuu- uuuu
HIRCC	●	●	●	---- 0001	---- 0001	---- 0001	---- uuuu
HXTC	●	●	●	---- -000	---- -000	---- -000	---- -uuu
LXTC	●	●		---- -000	---- -000	---- -000	---- -uuu
			●	---- --00	---- --00	---- --00	---- --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	0101 0101	0101 0101	uuuu uuuu
LVDC	●	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
MFI0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
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	●	●	●	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

Register	BH67F5250	BH67F5260	BH67F5270	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
PBPU	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	●	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	●	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PSCR	●	●	●	---- --00	---- --00	---- --00	---- --uu
TB0C	●	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
TB1C	●	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
SIMC0	●	●	●	1110 0000	1110 0000	1110 0000	uuuu uuuu
SIMC1(UMD=0)	●	●	●	1000 0001	1000 0001	1000 0001	uuuu uuuu
UUCR1* (UMD=1)	●	●	●	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
SIMA/SIMC2/UUCR2	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMD/UTXR_RXR	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMTOC (UMD=0)	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
UBRG* (UMD=1)	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
UUSR	●	●	●	0000 1011	0000 1011	0000 1011	uuuu uuuu
SPIAC0	●	●	●	111- --00	111- --00	111- --00	uuu- --uu
SPIAC1	●	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
SPIAD	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
EEA	●			-000 0000	-000 0000	-000 0000	-uuu uuuu
		●		0000 0000	0000 0000	0000 0000	uuuu uuuu
EEAL			●	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEAH			●	---- ---0	---- ---0	---- ---0	---- ---u
EED	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMC0	●	●	●	0000 0---	0000 0---	0000 0---	uuuu u---
STMC1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMRP	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0C0	●	●	●	0000 0---	0000 0---	0000 0---	uuuu u---
PTM0C1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DH	●	●	●	---- --00	---- --00	---- --00	---- --uu
PTM0AL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0AH	●	●	●	---- --00	---- --00	---- --00	---- --uu
PTM0RPL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	●	●	●	---- --00	---- --00	---- --00	---- --uu
MDUWR0	●	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR1	●	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR2	●	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR3	●	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR4	●	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR5	●	●	●	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWCTRL	●	●	●	00-- ----	00-- ----	00-- ----	uu-- ----

Register	BH67F5250	BH67F5260	BH67F5270	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
PE	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEPU	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADCS	•	•	•	---0 0000	---0 0000	---0 0000	---u uuuu
ADCR0	•	•	•	0010 00-0	0010 00-0	0010 00-0	uuuu uu-u
ADCR1	•	•	•	0000 000-	0000 000-	0000 000-	uuuu uuu-
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	•	•	•	--00 0000	--00 0000	--00 0000	--uu 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
DSDAH	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
DSDAL	•	•	•	---- 0000	---- 0000	---- 0000	---- uuuu
DSDACC	•	•	•	00-- ----	00-- ----	00-- ----	uu-- ----
DSOPC	•	•	•	---- 1010	---- 1010	---- 1010	---- uuuu
PD	•	•	•	-111 1111	-111 1111	-111 1111	-uuu uuuu
PDC	•	•	•	-111 1111	-111 1111	-111 1111	-uuu uuuu
PDPU	•	•	•	-000 0000	-000 0000	-000 0000	-uuu uuuu
LCDC0	•	•	•	0000 -000	0000 -000	0000 -000	uuuu -uuu
LCDCP	•	•	•	---- 0-00	---- 0-00	---- 0-00	---- u-uu
LCDC2	•	•	•	000- -000	000- -000	000- -000	uuu- -uuu
PF	•	•	•	-111 1111	-111 1111	-111 1111	-uuu uuuu
PFC	•	•	•	-111 1111	-111 1111	-111 1111	-uuu uuuu
PFPU	•	•	•	-000 0000	-000 0000	-000 0000	-uuu uuuu
PMPS	•	•	•	--00 0000	--00 0000	--00 0000	--uu uuuu
PTM1C0	•	•	•	0000 0---	0000 0---	0000 0---	uuuu u---
PTM1C1	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DH	•	•	•	---- --00	---- --00	---- --00	---- --uu
PTM1AL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1AH	•	•	•	---- --00	---- --00	---- --00	---- --uu
PTM1RPL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	•	•	•	---- --00	---- --00	---- --00	---- --uu
PTM2C0	•	•	•	0000 0---	0000 0---	0000 0---	uuuu u---
PTM2C1	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DH	•	•	•	---- --00	---- --00	---- --00	---- --uu
PTM2AL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2AH	•	•	•	---- --00	---- --00	---- --00	---- --uu
PTM2RPL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	•	•	•	---- --00	---- --00	---- --00	---- --uu
EEC	•	•	•	---- 0000	---- 0000	---- 0000	---- uuuu
FC0	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	BH67F5250	BH67F5260	BH67F5270	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
FC1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC2	●	●	●	---- ---0	---- ---0	---- ---0	---- ---u
FARL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	●			---0 0000	---0 0000	---0 0000	---u uuuu
		●		--00 0000	--00 0000	--00 0000	--uu uuuu
			●	-000 0000	-000 0000	-000 0000	-uuu uuuu
FD0L	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0H	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1L	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1H	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2L	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2H	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3L	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3H	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS0	●	●	●	00-- 0--0	00-- 0--0	00-- 0--0	uu-- u--u
IFS1	●	●	●	-000 ---0	-000 ---0	-000 ---0	-uuu ---u
PAS0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	●	●	●	0000 00--	0000 00--	0000 00--	uuuu uu--
PBS1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC2	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS1	●	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
PES0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PES1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PFS0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PFS1	●	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu

Note: “u” stands for unchanged
“x” stands for unknown
“-” stands for unimplemented
“*”: The UUCR1 and SIMC1 registers share the same memory address while the UBRG and SIMTOC registers share the same memory address. The default value of the UUCR1 or UBRG register can be obtained when the UMD bit is set high by application program after a reset.

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.

These devices provide bidirectional input/output lines labeled with port names PA~PF. 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
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
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
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	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	—	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	—	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	—	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PF	—	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	—	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
PFPU	—	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0

"—": 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 an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using registers PAPU~PFPU, 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 input or NMOS output. Otherwise, the pull-high resistors cannot be enabled.

• **PxPU Register**

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

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

Port A Wake-up

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

Note that the wake-up function can be controlled by the wake-up control registers only when the pin-shared functional pin is selected as general purpose input/output 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~PFC, 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, C, D, E and F. However, the actual available bits for each I/O port may be different.

I/O Port Source Current Selection

The source current of each pin in these 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	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
SLEDC2	SLEDC27	SLEDC26	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20

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: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

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

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

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

• **SLEDC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
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 **SLEDC17~SLEDC16**: PD6~PD4 source current selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 5~4 **SLEDC15~SLEDC14**: PD3~PD0 source current selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 3~2 **SLEDC13~SLEDC12**: PC7~PC4 source current selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 1~0 **SLEDC11~SLEDC10**: PC3~PC0 source current selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

• **SLEDC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC27	SLEDC26	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20
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 **SLEDC27~SLEDC26**: PF6~PF4 source current selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 5~4 **SLEDC25~SLEDC24**: PF3~PF0 source current selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 3~2 **SLEDC23~SLEDC22**: PE7~PE4 source current selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 1~0 **SLEDC21~SLEDC20**: PE3~PE0 source current selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

I/O Port Power Source Control

These devices support different I/O port power source selections for PA6~PA7, PB2~PB3 and PC4~PC5. 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**: PC5~PC4 pin power source selection
0x: VDD
1x: VDDIO

Bit 3~2 **PMPS3~PMPS2**: PB3~PB2 pin power source selection
0x: VDD
1x: VDDIO

Bit 1~0 **PMPS1~PMPS0**: PA7~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 device includes Port "x" output function Selection register "n", labeled as PxSn, and Input Function Selection register "n", labeled as IFSi, which can select the desired functions of the multi-function pin-shared pins.

When the pin-shared input function is selected to be used, the corresponding input and output functions selection should be properly managed. For example, if the I^2C SDA line is used, the corresponding output pin-shared function should be configured as the SDI/SDA function by configuring the PxSn register and the SDA signal input should be properly selected using the IFSi register. However, if the external interrupt function is selected to be used, the relevant output pin-shared function should be selected as an I/O function and the interrupt input signal should be selected.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control

register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as INTn, xTCKn, xTPnI, 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
IFS0	PTP2IPS	PTP1IPS	—	—	PTCK2PS	—	—	STCKPS
IFS1	—	SCSAPS	SDIAPS	SCKAPS	—	—	—	RXPS
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	—	—
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
PDS1	—	—	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
PES0	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
PES1	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
PFS0	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
PFS1	—	—	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10

Pin-shared Function Selection Register List

• PAS0 Register

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

Bit 7~6 **PAS07~PAS06**: PA3 Pin-Shared function selection
00: PA3/PTP1I
01: PTP1
10: SDOA
11: OSC2

Bit 5~4 **PAS05~PAS04**: PA2 Pin-Shared function selection
00: PA2/PTCK0
01: PA2/PTCK0
10: PA2/PTCK0
11: XT2

Bit 3~2 **PAS03~PAS02**: PA1 Pin-Shared function selection
00: PA1/INT0/STCK
01: PA1/INT0/STCK
10: LVDIN
11: PTP0B

Bit 1~0 **PAS01~PAS00**: PA0 Pin-Shared function selection
00: PA0/PTP0I
01: PA0/PTP0I
10: PTP0
11: XT1

• **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/PTCK2
01: PA7/PTCK2
10: PA7/PTCK2
11: SDI/SDA/RX

Bit 5~4 **PAS15~PAS14**: PA6 Pin-Shared function selection
00: PA6/STCK
01: SDIA
10: SCK/SCL
11: PA6/STCK

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

Bit 1~0 **PAS11~PAS10**: PA4 Pin-Shared function selection
00: PA4/PTP2I
01: PTP2
10: $\overline{\text{SCSA}}$
11: OSC1

• **PBS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	—	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	—
POR	0	0	0	0	0	0	—	—

Bit 7~6 **PBS07~PBS06**: PB3 Pin-Shared function selection
00: PB3
01: PB3
10: SDIA
11: SDO/TX

Bit 5~4 **PBS05~PBS04**: PB2 Pin-Shared function selection
00: PB2
01: PB2
10: $\overline{\text{SCS}}$
11: SCKA

Bit 3~2 **PBS03~PBS02**: PB1 Pin-Shared function selection
00: PB1/INT1
01: PB1/INT1
10: STPB
11: PB1/INT1

Bit 1~0 Unimplemented, read as "0"

• **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
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 **PBS17~PBS16**: PB7 Pin-Shared function selection
00: PB7
01: SEG12
10: PB7
11: PB7

Bit 5~4 **PBS15~PBS14**: PB6 Pin-Shared function selection
00: PB6
01: SEG13
10: PB6
11: PB6

Bit 3~2 **PBS13~PBS12**: PB5 Pin-Shared function selection
00: PB5
01: SEG14
10: PB5
11: PB5

Bit 1~0 **PBS11~PBS10**: PB4 Pin-Shared function selection
00: PB4
01: SEG15
10: PB4
11: PB4

• **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
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 **PCS07~PCS06**: PC3 Pin-Shared function selection
00: PC3/PTCK2
01: SEG18
10: PC3/PTCK2
11: PC3/PTCK2

Bit 5~4 **PCS05~PCS04**: PC2 Pin-Shared function selection
00: PC2/PTP2I
01: PTP2
10: SEG19
11: PC2/PTP2I

Bit 3~2 **PCS03~PCS02**: PC1 Pin-Shared function selection
00: PC1/PTCK1
01: SEG20
10: PC1/PTCK1
11: PC1/PTCK1

Bit 1~0 **PCS01~PCS00**: PC0 Pin-Shared function selection
00: PC0/PTP1I
01: PC0/PTP1I
10: PTP1
11: PC0/PTP1I

• **PCS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
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 **PCS17~PCS16**: PC7 Pin-Shared function selection
00: PC7
01: SEG16
10: PC7
11: PC7

Bit 5~4 **PCS15~PCS14**: PC6 Pin-Shared function selection
00: PC6
01: SEG17
10: PC6
11: PC6

Bit 3~2 **PCS13~PCS12**: PC5 Pin-Shared function selection
00: PC5
01: PC5
10: SDI/SDA/RX
11: SDOA

Bit 1~0 **PCS11~PCS10**: PC4 Pin-Shared function selection
00: PC4
01: PC4
10: SDO/TX
11: $\overline{\text{SCSA}}$

• **PDS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
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 **PDS07~PDS06**: PD3 Pin-Shared function selection
00: PD3
01: SEG3
10: COM7
11: PD3

Bit 5~4 **PDS05~PDS04**: PD2 Pin-Shared function selection
00: PD2
01: SEG2
10: COM6
11: PD2

Bit 3~2 **PDS03~PDS02**: PD1 Pin-Shared function selection
00: PD1
01: SEG1
10: COM5
11: PD1

Bit 1~0 **PDS01~PDS00**: PD0 Pin-Shared function selection
00: PD0
01: SEG0
10: COM4
11: PD0

• **PDS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
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 5~4 **PDS15~PDS14**: PD6 Pin-Shared function selection
00: PD6
01: SEG27
10: V2
11: PD6

Bit 3~2 **PDS13~PDS12**: PD5 Pin-Shared function selection
00: PD5
01: SEG26
10: C2
11: PD5

Bit 1~0 **PDS11~PDS10**: PD4 Pin-Shared function selection
00: PD4
01: SEG25
10: C1
11: PD4

• **PES0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
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 **PES07~PES06**: PE3 Pin-Shared function selection
00: PE3
01: SEG8
10: AN12
11: PE3

Bit 5~4 **PES05~PES04**: PE2 Pin-Shared function selection
00: PE2
01: SEG9
10: AN13
11: PE2

Bit 3~2 **PES03~PES02**: PE1 Pin-Shared function selection
00: PE1
01: SEG10
10: AN14
11: PE1

Bit 1~0 **PES01~PES00**: PE0 Pin-Shared function selection
00: PE0
01: SEG11
10: AN15
11: PE0

• **PES1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
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 **PES17~PES16**: PE7 Pin-Shared function selection
00: PE7
01: SEG4
10: AN8
11: PE7

Bit 5~4 **PES15~PES14**: PE6 Pin-Shared function selection
00: PE6
01: SEG5
10: AN9
11: PE6

Bit 3~2 **PES13~PES12**: PE5 Pin-Shared function selection
00: PE5
01: SEG6
10: AN10
11: PE5

Bit 1~0 **PES11~PES10**: PE4 Pin-Shared function selection
00: PE4
01: SEG7
10: AN11
11: PE4

• **PFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
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 **PFS07~PFS06**: PF3 Pin-Shared function selection
00: PF3
01: SEG24
10: AN7
11: PF3

Bit 5~4 **PFS05~PFS04**: PF2 Pin-Shared function selection
00: PF2
01: SEG23
10: AN6
11: PF2

Bit 3~2 **PFS03~PFS02**: PF1 Pin-Shared function selection
00: PF1
01: Reserved
10: AN5
11: OPIP

Bit 1~0 **PFS01~PFS00**: PF0 Pin-Shared function selection
00: PF0
01: Reserved
10: AN4
11: OPIN

• **PFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10
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 **PFS15~PFS14**: PF6 Pin-Shared function selection
00: PF6
01: SEG28
10: PF6
11: PF6

Bit 3~2 **PFS13~PFS12**: PF5 Pin-Shared function selection
00: PF5
01: SEG29
10: PF5
11: PF5

Bit 1~0 **PFS11~PFS10**: PF4 Pin-Shared function selection
00: PF4
01: VDDIO
10: PF4
11: PF4

• **IFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTP2IPS	PTP1IPS	—	—	PTCK2PS	—	—	STCKPS
R/W	R/W	R/W	—	—	R/W	—	—	R/W
POR	0	0	—	—	0	—	—	0

Bit 7 **PTP2IPS**: PTP2I input source pin selection
0: PTP2I on PA4
1: PTP2I on PC2

Bit 6 **PTP1IPS**: PTP1I input source pin selection
0: PTP1I on PA3
1: PTP1I on PC0

Bit 5~4 Unimplemented, read as "0"

Bit 3 **PTCK2PS**: PTCK2 input source pin selection
0: PTCK2 on PC3
1: PTCK2 on PA7

Bit 2~1 Unimplemented, read as "0"

Bit 0 **STCKPS**: STCK input source pin selection
0: STCK on PA1
1: STCK on PA6

• **IFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	SCSAPS	SDIAPS	SCKAPS	—	—	—	RXPS
R/W	—	R/W	R/W	R/W	—	—	—	R/W
POR	—	0	0	0	—	—	—	0

Bit 7 Unimplemented, read as "0"

Bit 6 **SCSAPS**: $\overline{\text{SCSA}}$ input source pin selection
0: $\overline{\text{SCSA}}$ on PA4
1: $\overline{\text{SCSA}}$ on PC4

Bit 5 **SDIAPS**: SDIA input source pin selection
0: SDIA on PA6
1: SDIA on PB3

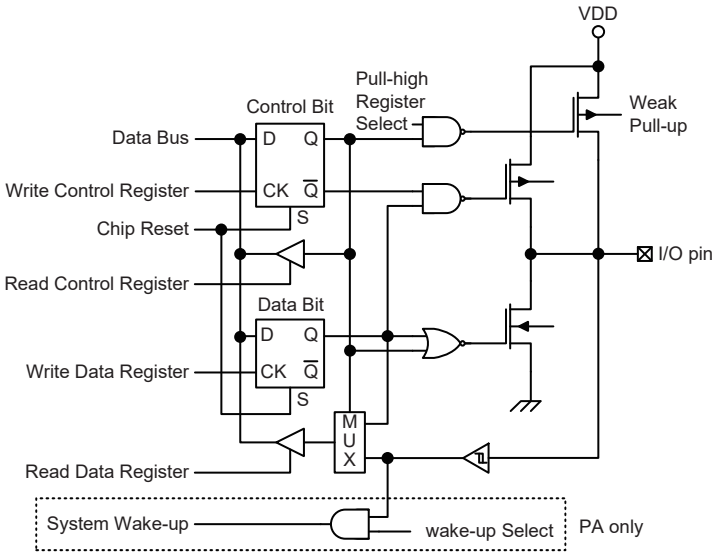
Bit 4 **SCKAPS**: SCKA input source pin selection
0: SCKA on PA5
1: SCKA on PB2

Bit 3~1 Unimplemented, read as "0"

Bit 0 **RXPS**: RX input source pin selection
0: RX on PA7
1: RX on PC5

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 diagram, it is supplied as a guide only to assist with the functional understanding of the logic function I/O pins. The wide range of pin-shared structures does not permit all types to be shown.



Logic Function Input/Output Structure

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

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

Introduction

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

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

TM Function Summary

Device	STM	PTM0	PTM1	PTM2
BH67F5250 BH67F5260 BH67F5270	16-bit STM	10-bit PTM	10-bit PTM	10-bit PTM

TM Name/Type Reference

TM Operation

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

TM Clock Source

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

TM Interrupts

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

TM External Pins

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

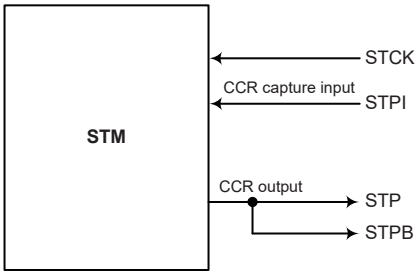
The other xTMn input pin, xTPnI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the xTnIO1~xTnIO0 bits in the xTMnC1 register. There is another capture input, PTCKn, for PTMn capture input mode, which can be used as the external trigger input source except the PTPnI pin.

The TMs each have one output pin with the label xTPn while some TMs have an additionl output pin with the lebel xTPnB. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTPn and xTPnB output pins are also the pins where the TM generates the PWM output waveform.

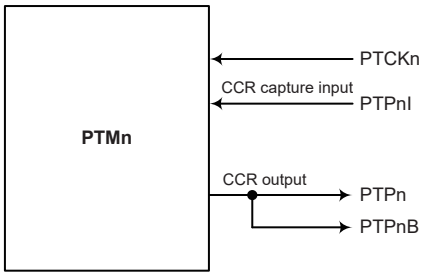
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using the relevant pin-shared function selection bits described in the Pin-shared Function section. The details of the pin-shared function selection are described in the pin-shared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

Device	STM		PTM0		PTM1		PTM2	
	Input	Output	Input	Output	Input	Output	Input	Output
BH67F5250 BH67F5260 BH67F5270	STCK, STPI	STP, STPB	PTCK0, PTP0I	PTP0, PTP0B	PTCK1, PTP1I	PTP1	PTCK2, PTP2I	PTP2

TM External Pins



STM Function Pin Block Diagram



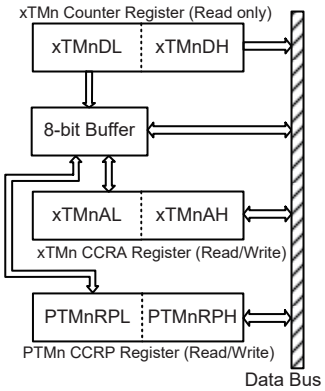
Note: Only the PTM0 has the inverted output pin PTP0B.

PTMn Function Pin Block Diagram (n=0~2)

Programming Considerations

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

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



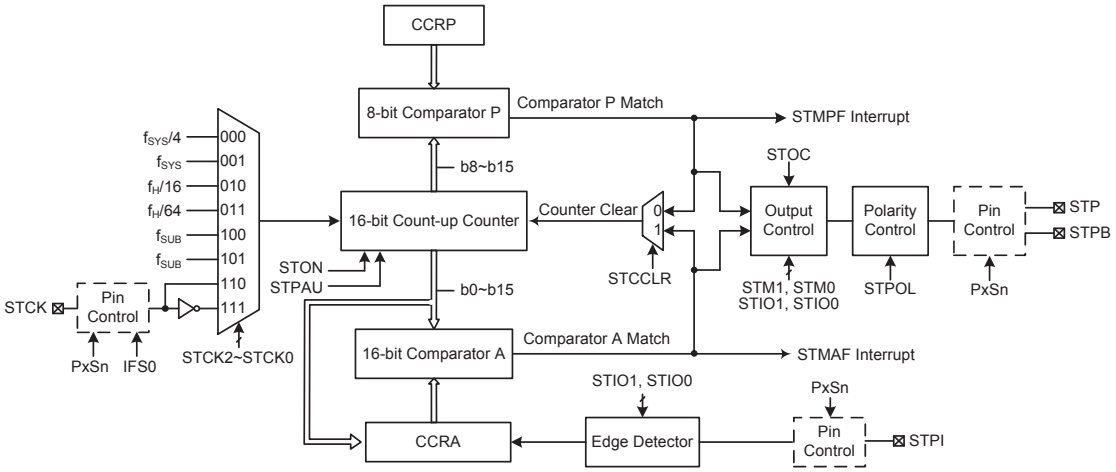
The following steps show the read and write procedures:

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

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
BH67F5250 BH67F5260 BH67F5270	16-bit STM	STCK, STPI	STP, STPB



Standard Type TM Block Diagram

Standard TM Operation

The size of Standard TM is 16-bit wide and its core is a 16-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP comparator is 8-bit wide whose value is compared the with highest 8 bits in the counter while the CCRA is the sixteen bits and therefore compares all counter bits.

The only way of changing the value of the 16-bit counter using the application program, is to clear the counter by changing the 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 16-bit value, while a read/write register pair exists to store the internal 16-bit CCRA value. The STMRP register is used to store the 8-bit CCRP value. The remaining two registers are control registers which setup the different operating and control modes.

Register Name	Bit							
	7	6	5	4	3	2	1	0
STMC0	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
STMC1	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
STMDL	D7	D6	D5	D4	D3	D2	D1	D0
STMDH	D15	D14	D13	D12	D11	D10	D9	D8
STMAL	D7	D6	D5	D4	D3	D2	D1	D0
STMAH	D15	D14	D13	D12	D11	D10	D9	D8
STMRP	STRP7	STRP6	STRP5	STRP4	STRP3	STRP2	STRP1	STRP0

16-bit Standard TM Register List

- **STMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	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**: Select STM Counter clock
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 Unimplemented, read as “0”

• **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**: Select STM Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

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

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

Capture Input Mode
00: Input capture at rising edge of 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 setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the STM output pin should be setup 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 16-bit Counter bit 7 ~ bit 0

• STMDH Register

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

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

• **STMAH Register**

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

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

• **STMRP Register**

Bit	7	6	5	4	3	2	1	0
Name	STRP7	STRP6	STRP5	STRP4	STRP3	STRP2	STRP1	STRP0
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 **STRP7~STRP0**: STM CCRP 8-bit register, compared with the STM counter bit 15 ~ bit 8
Comparator P Match Period=
0: 65536 STM clocks
1~255: (1~255)×256 STM clocks
These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight 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 eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

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

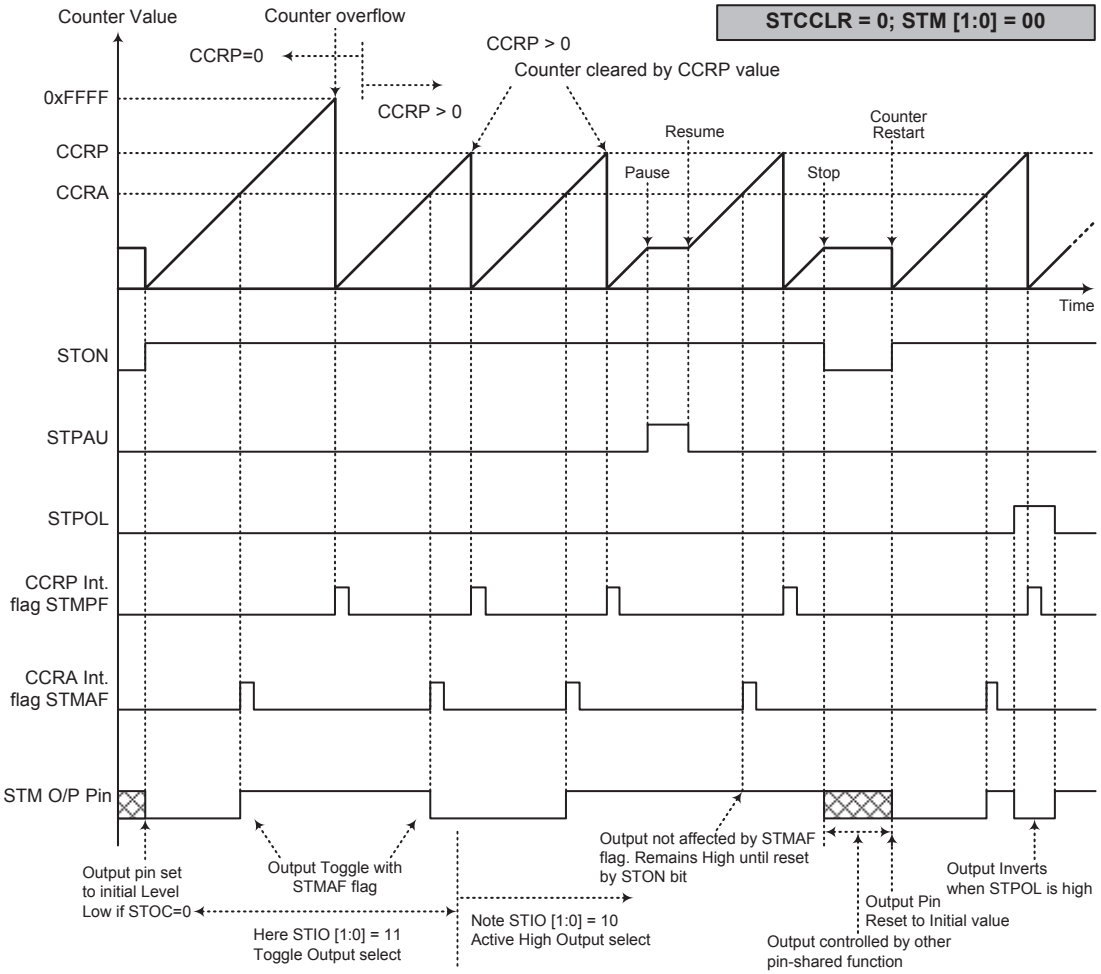
To select this mode, bits STM1 and STM0 in the STMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the STCCLR 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 clear to 0.

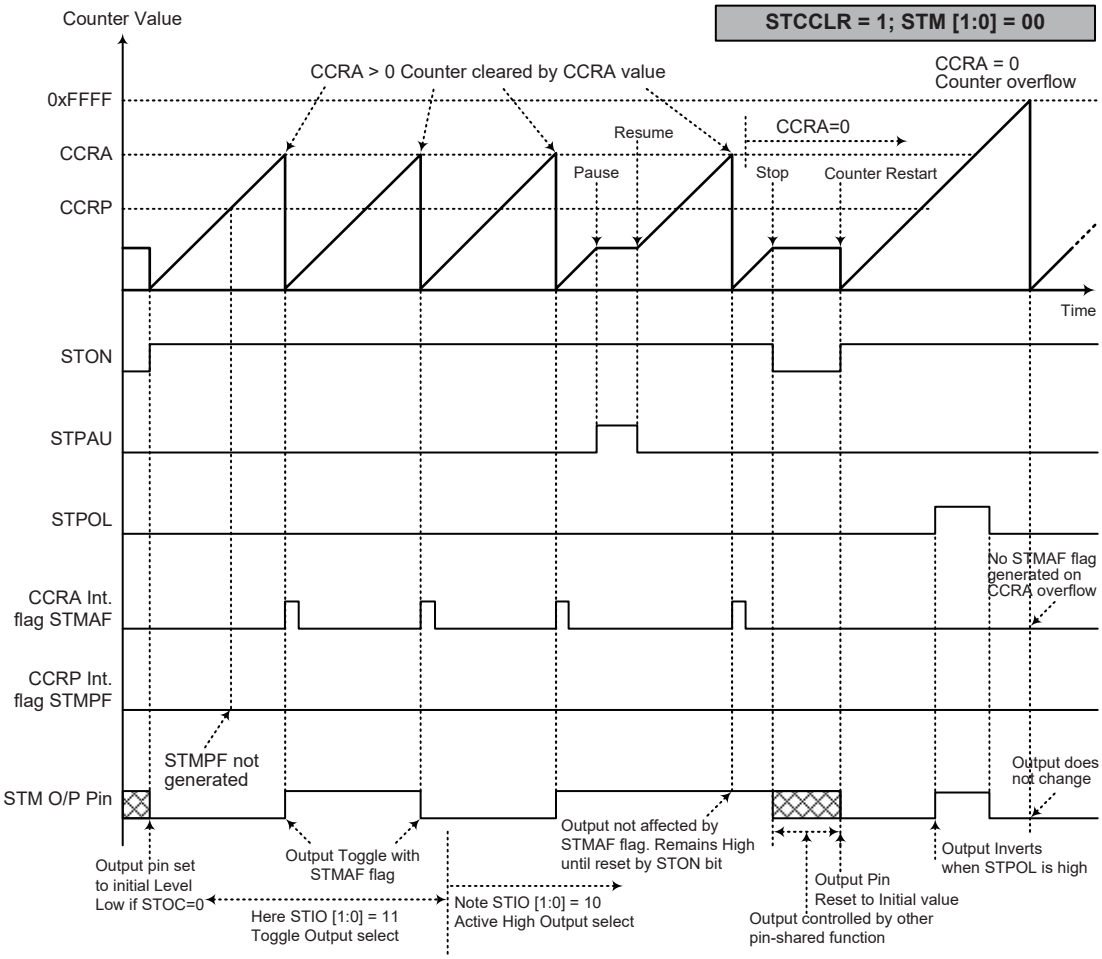
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 16-bit, FFFF Hex, value, however here the STMAF interrupt request flag will not be generated.

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 setup after the STON bit changes from low to high, is setup 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.

• 16-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=0

CCRP	1~255	0
Period	CCRP×256	65536
Duty	CCRA	

If f_{SYS}=16MHz, STM clock source is f_{SYS}/4, CCRP=2 and CCRA=128,

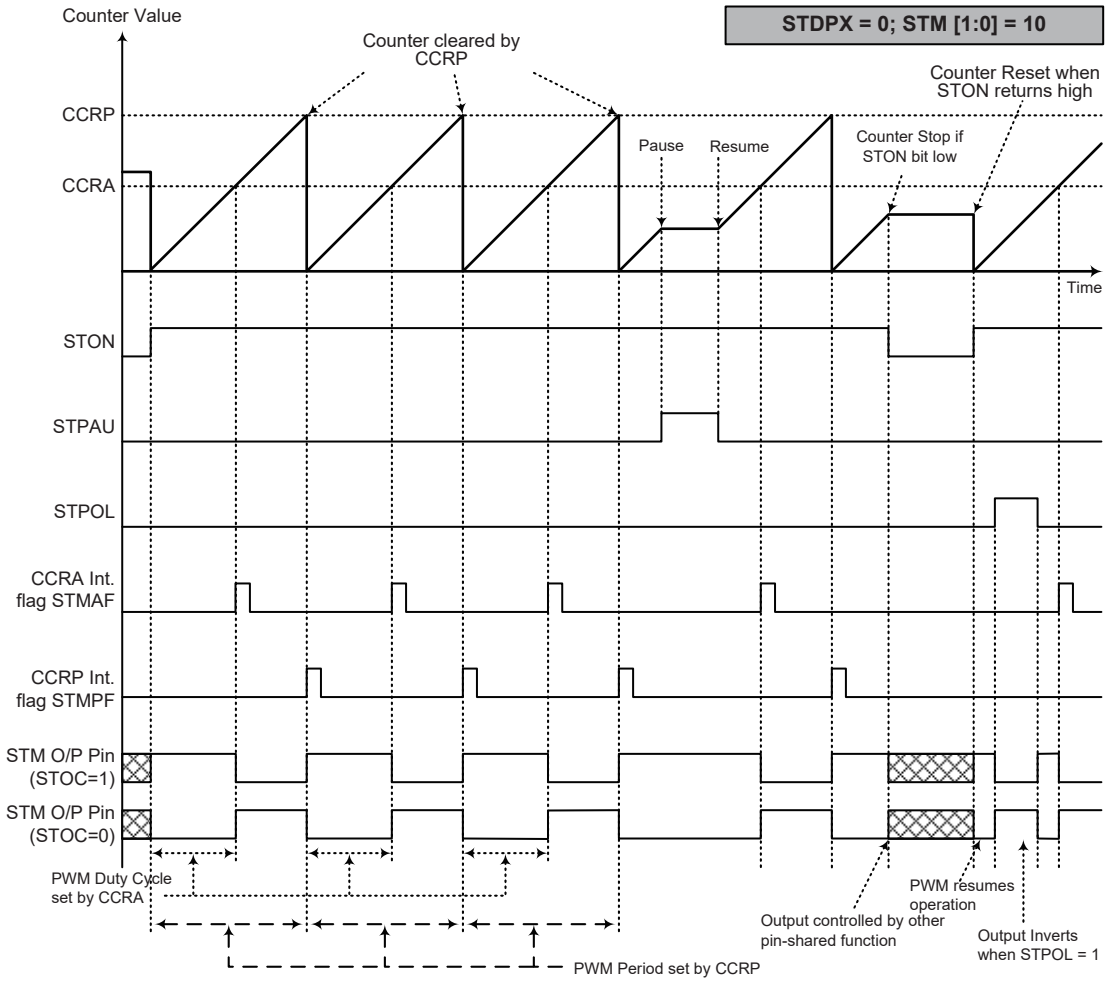
The STM PWM output frequency=(f_{SYS}/4)/(2×256)=f_{SYS}/2048=7.8125kHz, duty=128/(2×256)=25%.

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%.

• 16-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=1

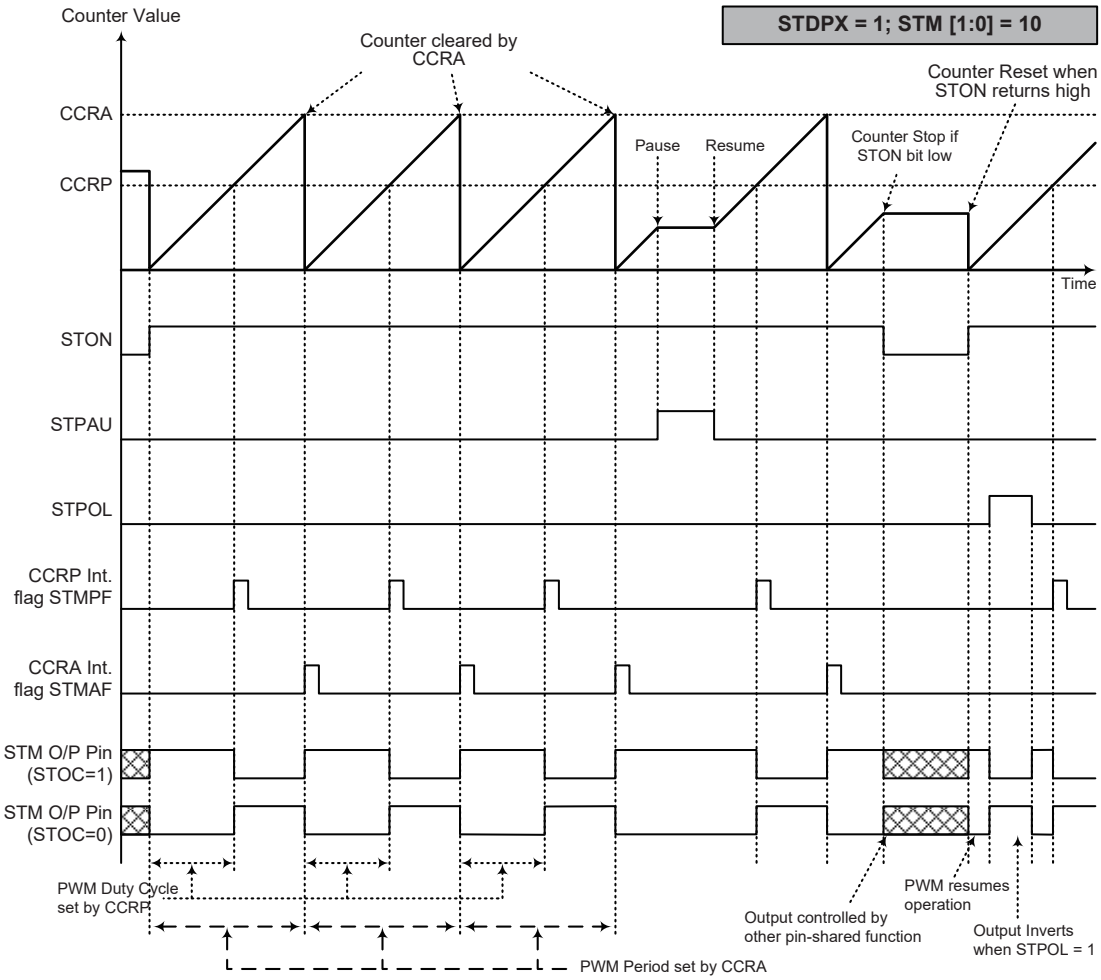
CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

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

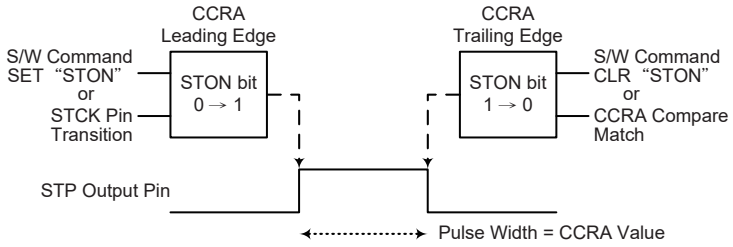
Note: 1. Here STDPX=1 – Counter cleared by CCRA
2. A counter clear sets the PWM Period
3. The internal PWM function continues even when STIO[1:0]=00 or 01
4. The STCCLR bit has no influence on PWM operation

Single Pulse Output Mode

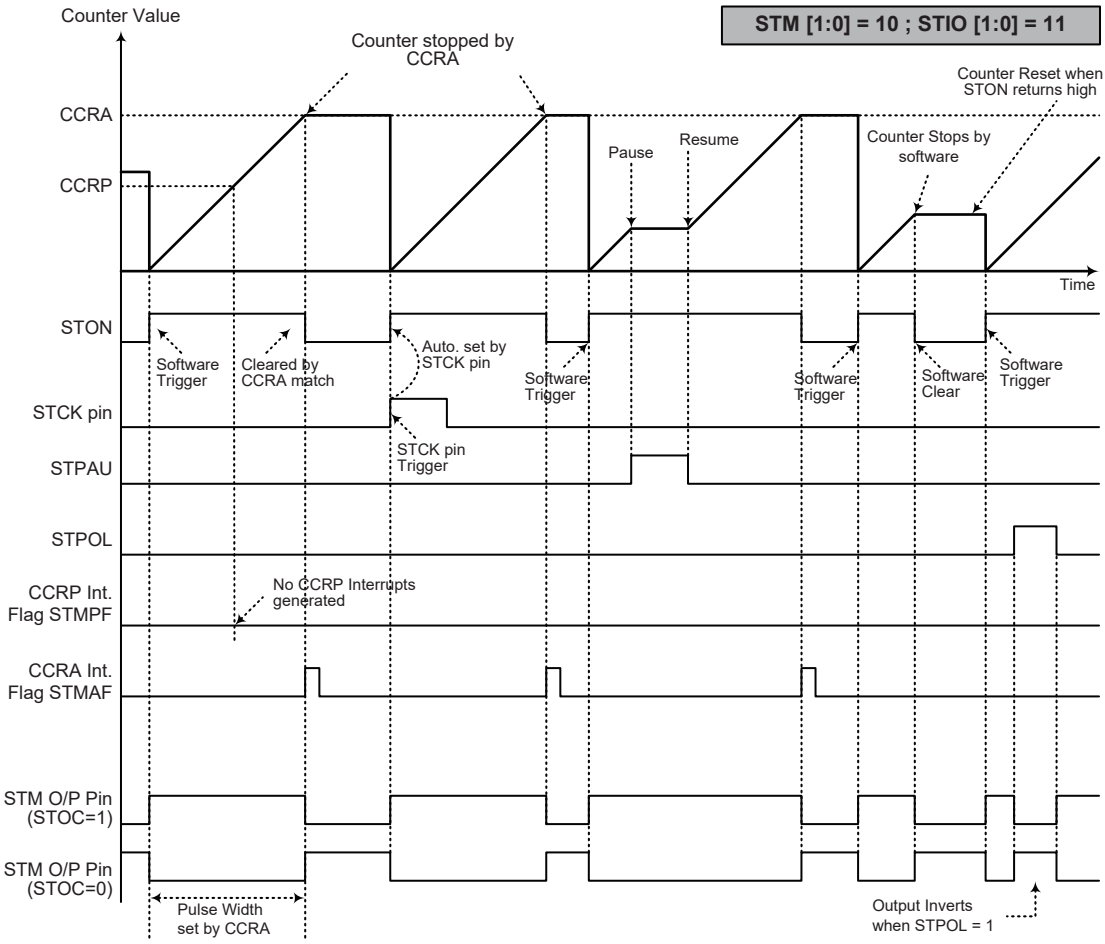
To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

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



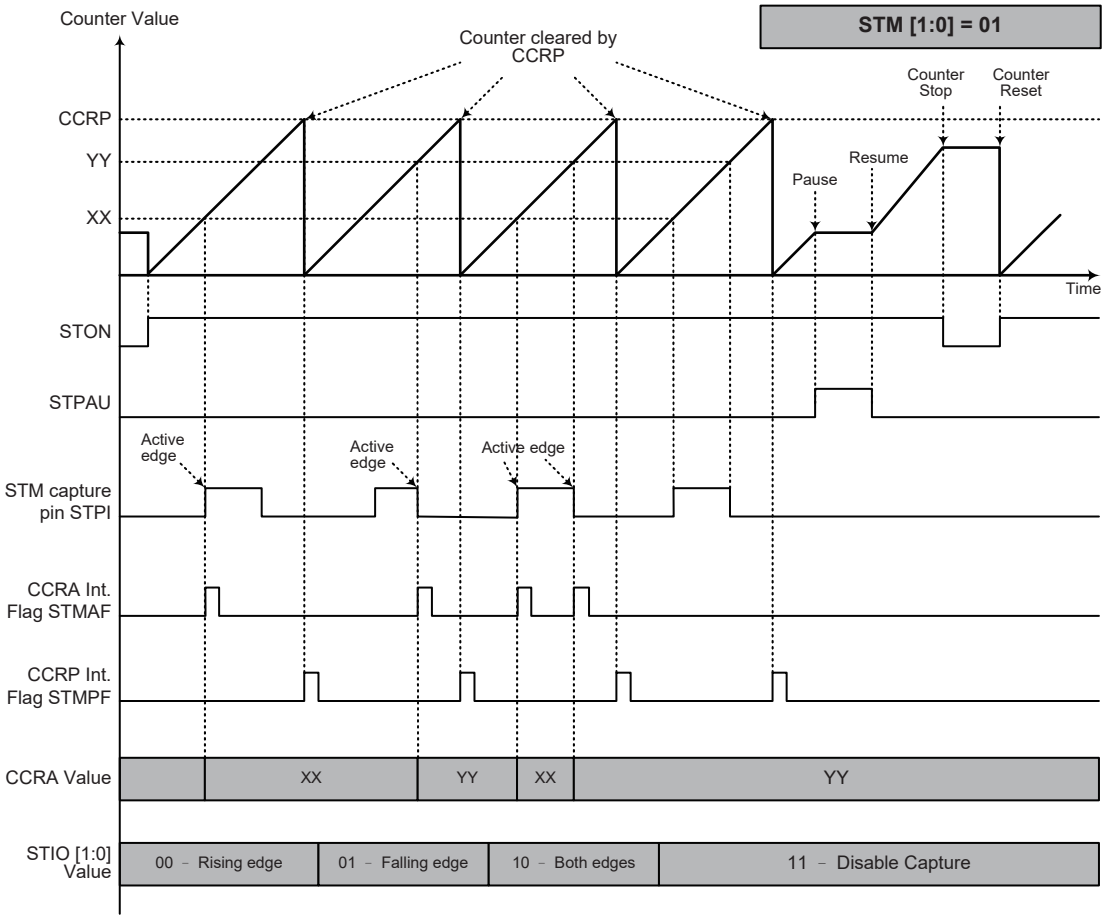
Single Pulse Output Mode

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.



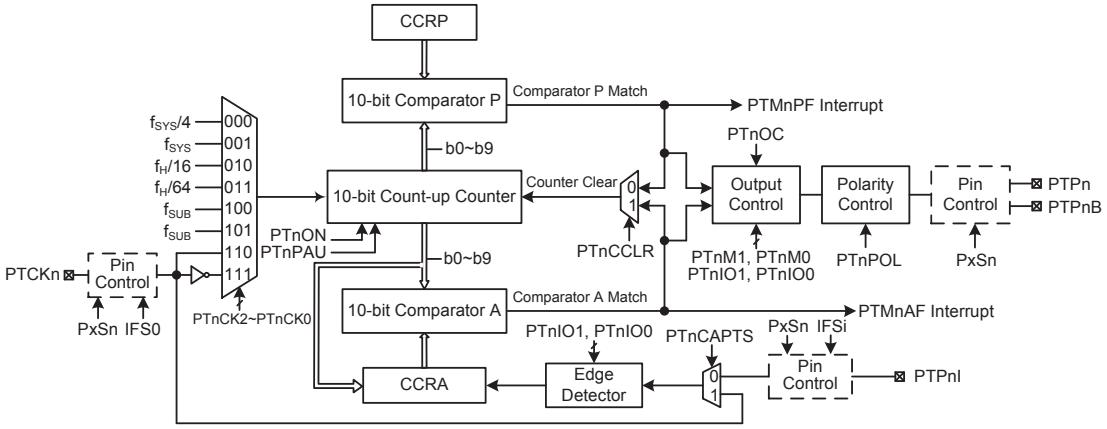
Capture Input Mode

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.

Periodic Type TM – PTM

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

Device	PTM Core	PTM Input Pin	PTM Output Pin
BH67F5250 BH67F5260 BH67F5270	10-bit PTM (PTM0, PTM1, PTM2)	PTCK0, PTP0I PTCK1, PTP1I PTCK2, PTP2I	PTP0, PTP0B PTP1 PTP2



Note: 1. The PTPnB is the inverted output of the PTPn and is only available for PTM0.
2. The external clock input source being selected using the IFS0 register is only available for PTM2.
3. The external PTPnI capture input source being selected using the IFS0 register is only available for PTM1 and PTM2.

Periodic Type TM Block Diagram (n=0~2)

Periodic TM Operation

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

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the PTnON 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 PTMn interrupt signal will also usually be generated. The Periodic Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control the output pins. All operating setup conditions are selected using relevant internal registers.

Periodic Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnRPL	PTnRP7	PTnRP6	PTnRP5	PTnRP4	PTnRP3	PTnRP2	PTnRP1	PTnRP0
PTMnRPH	—	—	—	—	—	—	PTnRP9	PTnRP8

10-bit Periodic TM Register List (n=0~2)

- **PTMnC0 Register (n=0~2)**

Bit	7	6	5	4	3	2	1	0
Name	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTnPAU**: PTMn 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 PTMn 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 **PTnCK2~PTnCK0**: Select PTMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: PTCKn rising edge clock
111: PTCKn falling edge clock

These three bits are used to select the clock source for the PTMn. 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 **PTnON**: PTMn Counter On/Off Control
0: Off
1: On

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

Bit 2~0 Unimplemented, read as "0"

• **PTMnC1 Register (n=0~2)**

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
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 **PTnM1~PTnM0**: Select PTMn Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the PTMn. To ensure reliable operation the PTMn should be switched off before any changes are made to the PTnM1 and PTnM0 bits. In the Timer/Counter Mode, the PTMn output pin state is undefined.

Bit 5~4 **PTnIO1~PTnIO0**: Select PTMn external pin function

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

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

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

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn output pin changes state when a compare match occurs from the Comparator A. The PTMn output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the PTMn output pin should be setup using the PTnOC bit in the PTMnC1 register. Note that the output level requested by the PTnIO1 and PTnIO0 bits must be different from the initial value setup using the PTnOC bit otherwise no change will occur on the PTMn output pin when a compare match occurs. After the PTMn output pin changes state, it can be reset to its initial level by changing the level of the PTnON bit from low to high.

In the PWM Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn 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 PTnIO1 and PTnIO0 bits only after the TM has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the PTMn is running.

Bit 3 **PTnOC**: PTMn PTPn Output control bit

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

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

This is the output control bit for the PTMn output pin. Its operation depends upon whether PTMn 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 PTMn is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the PTMn 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 PTMn output pin when the PTnON bit changes from low to high.

Bit 2 **PTnPOL**: PTMn PTPn Output polarity Control
0: Non-invert
1: Invert

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

Bit 1 **PTnCAPTS**: PTMn Capture Trigger Source Selection
0: From PTPnI pin
1: From PTCKn pin

Bit 0 **PTnCCLR**: Select PTMn Counter clear condition
0: PTMn Comparator P match
1: PTMn Comparator A match

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

• PTMnDL Register (n=0~2)

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

• PTMnDH Register (n=0~2)

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**: PTMn Counter High Byte Register bit 1 ~ bit 0
PTMn 10-bit Counter bit 9 ~ bit 8

• **PTMnAL Register (n=0~2)**

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

• **PTMnAH Register (n=0~2)**

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**: PTMn CCRA High Byte Register bit 1 ~ bit 0
PTMn 10-bit CCRA bit 9 ~ bit 8

• **PTMnRPL Register (n=0~2)**

Bit	7	6	5	4	3	2	1	0
Name	PTnRP7	PTnRP6	PTnRP5	PTnRP4	PTnRP3	PTnRP2	PTnRP1	PTnRP0
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 **PTnRP7~PTnRP0**: PTMn CCRP Low Byte Register bit 7 ~ bit 0
PTMn 10-bit CCRP bit 7 ~ bit 0

• **PTMnRPH Register (n=0~2)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PTnRP9	PTnRP8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PTnRP9~PTnRP8**: PTMn CCRP High Byte Register bit 1 ~ bit 0
PTMn 10-bit CCRP bit 9 ~ bit 8

Periodic Type TM Operating Modes

The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTnM1 and PTnM0 bits in the PTMnC1 register.

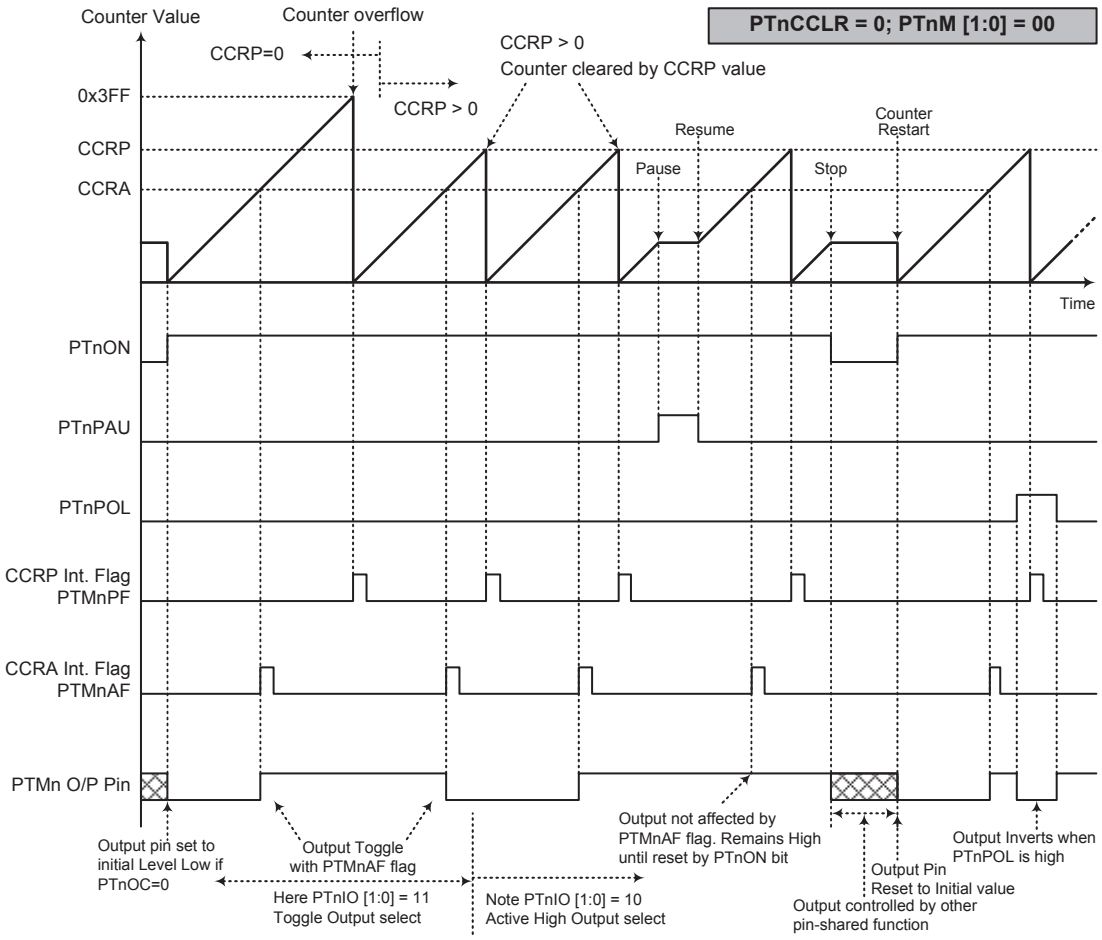
Compare Match Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 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 PTnCCLR 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 PTMnAF and PTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the PTnCCLR bit in the PTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is high no PTMnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be cleared to zero.

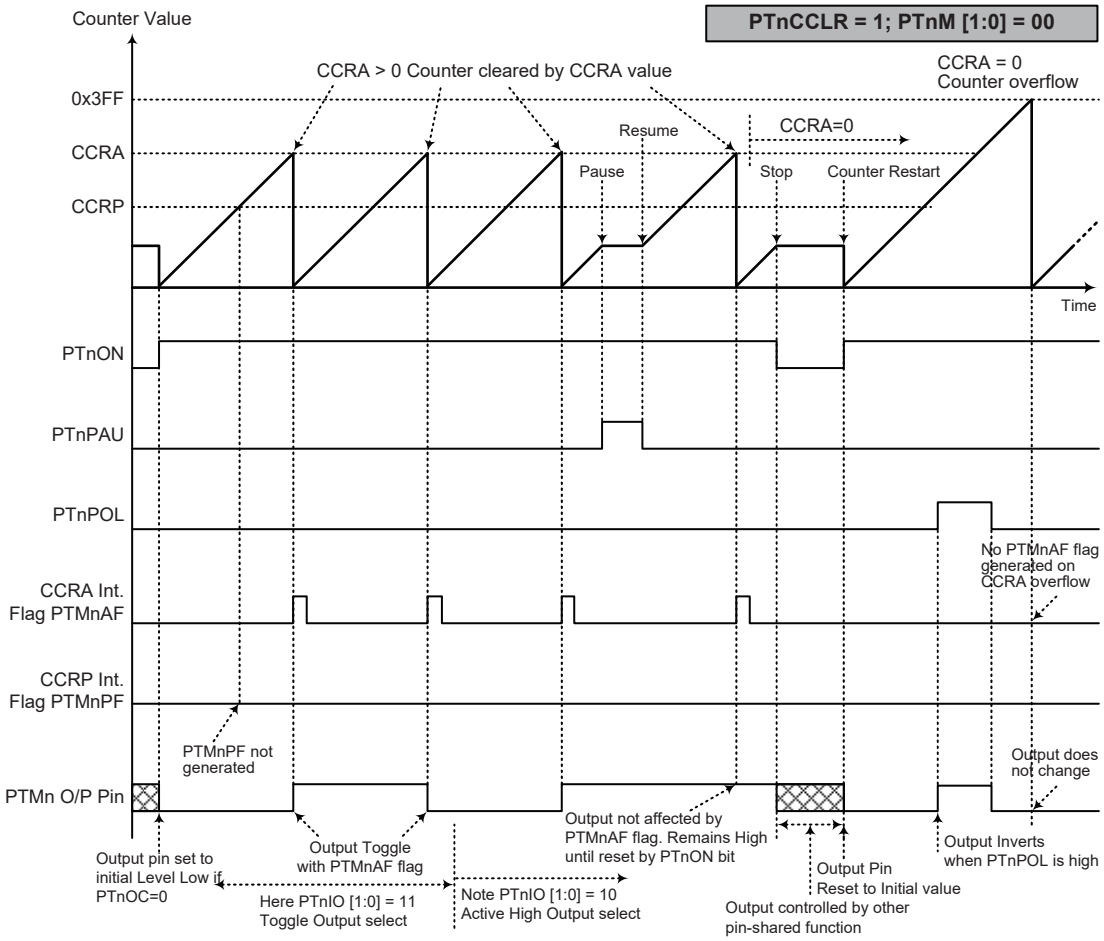
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the PTMnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the PTMn output pin, will change state. The PTMn output pin condition however only changes state when a PTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the PTMn output pin. The way in which the PTMn output pin changes state are determined by the condition of the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The PTMn output pin can be selected using the PTnIO1 and PTnIO0 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 PTMn output pin, which is setup after the PTnON bit changes from low to high, is setup using the PTnOC bit. Note that if the PTnIO1 and PTnIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – PTnCCLR=0 (n=0~2)

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



Compare Match Output Mode – PTnCCLR=1 (n=0~2)

Note: 1. With PTnCCLR=1 a Comparator A match will clear the counter
2. The PTMn output pin is controlled only by the PTMnAF flag
3. The output pin is reset to its initial state by a PTnON bit rising edge
4. The PTMnPF flag is not generated when PTnCCLR=1

Timer/Counter Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the TM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the TM 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 PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 10 respectively. The PWM function within the PTMn 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 PTMn 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 PTnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The PTnOC bit in the PTMnC1 register is used to select the required polarity of the PWM waveform while the two PTnIO1 and PTnIO0 bits are used to enable the PWM output or to force the PTMn output pin to a fixed high or low level. The PTnPOL bit is used to reverse the polarity of the PWM output waveform.

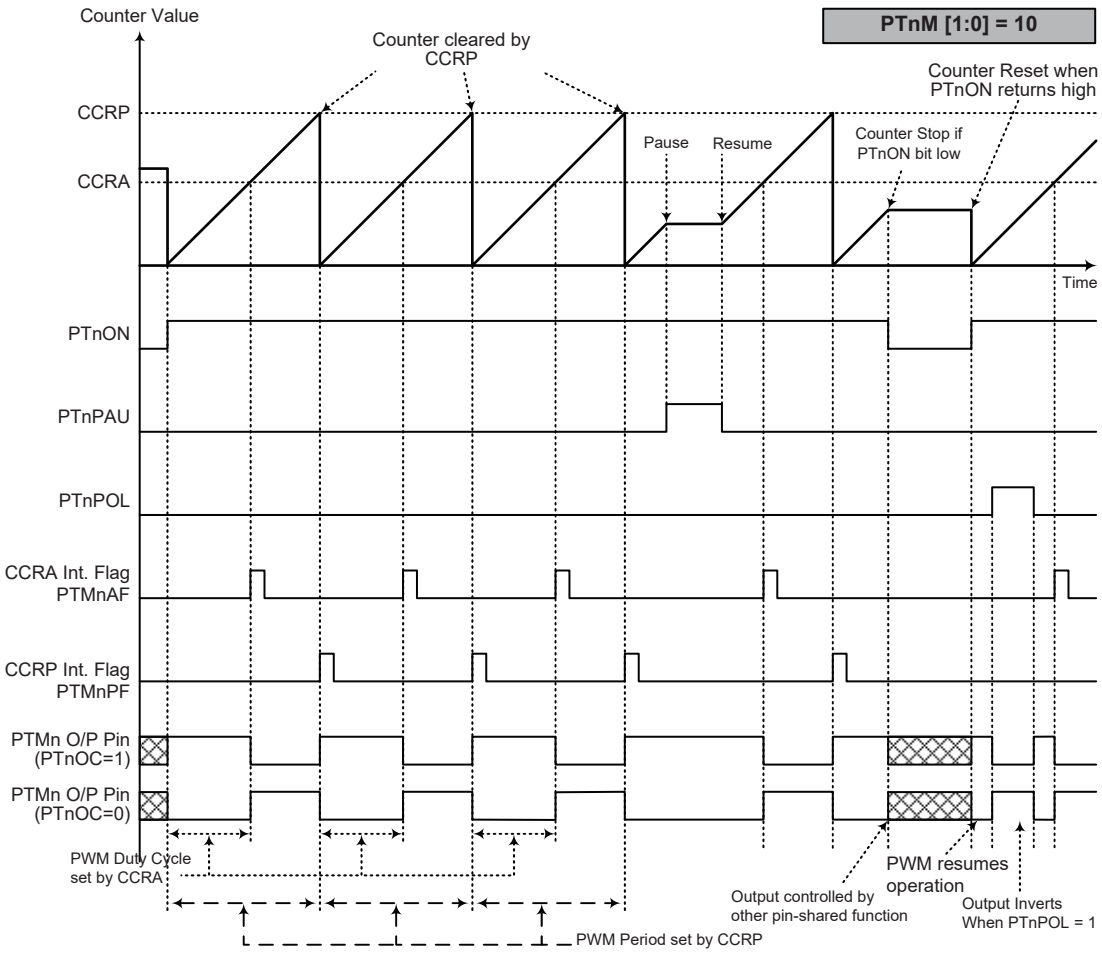
- **10-bit PTMn, PWM Output Mode, Edge-aligned Mode**

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

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

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

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



PWM Output Mode (n=0~2)

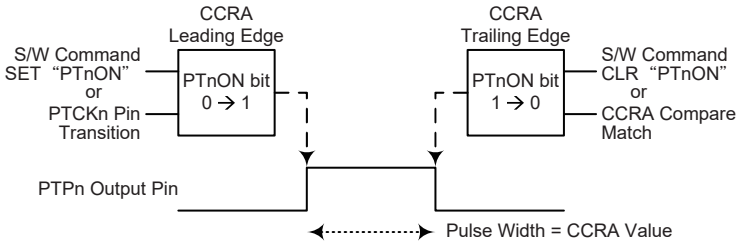
Note: 1. Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTnIO[1:0]=00 or 01
4. The PTnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

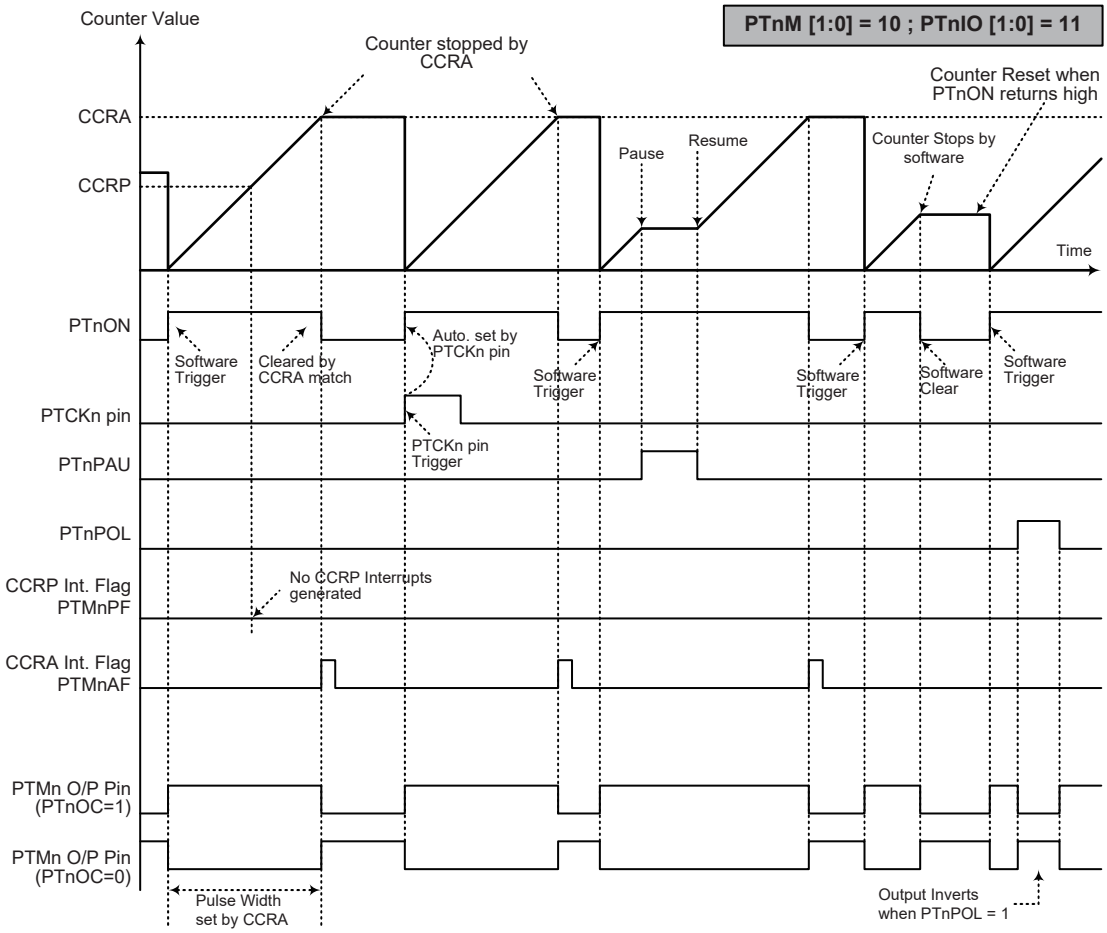
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

The trigger for the pulse output leading edge is a low to high transition of the PTnON bit, which can be implemented using the application program. However in the Single Pulse Output Mode, the PTnON bit can also be made to automatically change from low to high using the external PTCKn pin, which will in turn initiate the Single Pulse output. When the PTnON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The PTnON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the PTnON 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 PTnON 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 PTMn interrupt. The counter can only be reset back to zero when the PTnON bit changes from low to high when the counter restarts. In the Single Pulse Output Mode CCRP is not used. The PTnCCLR bit is not used in this Mode.



Single Pulse Generation (n=0~2)



Single Pulse Output Mode (n=0~2)

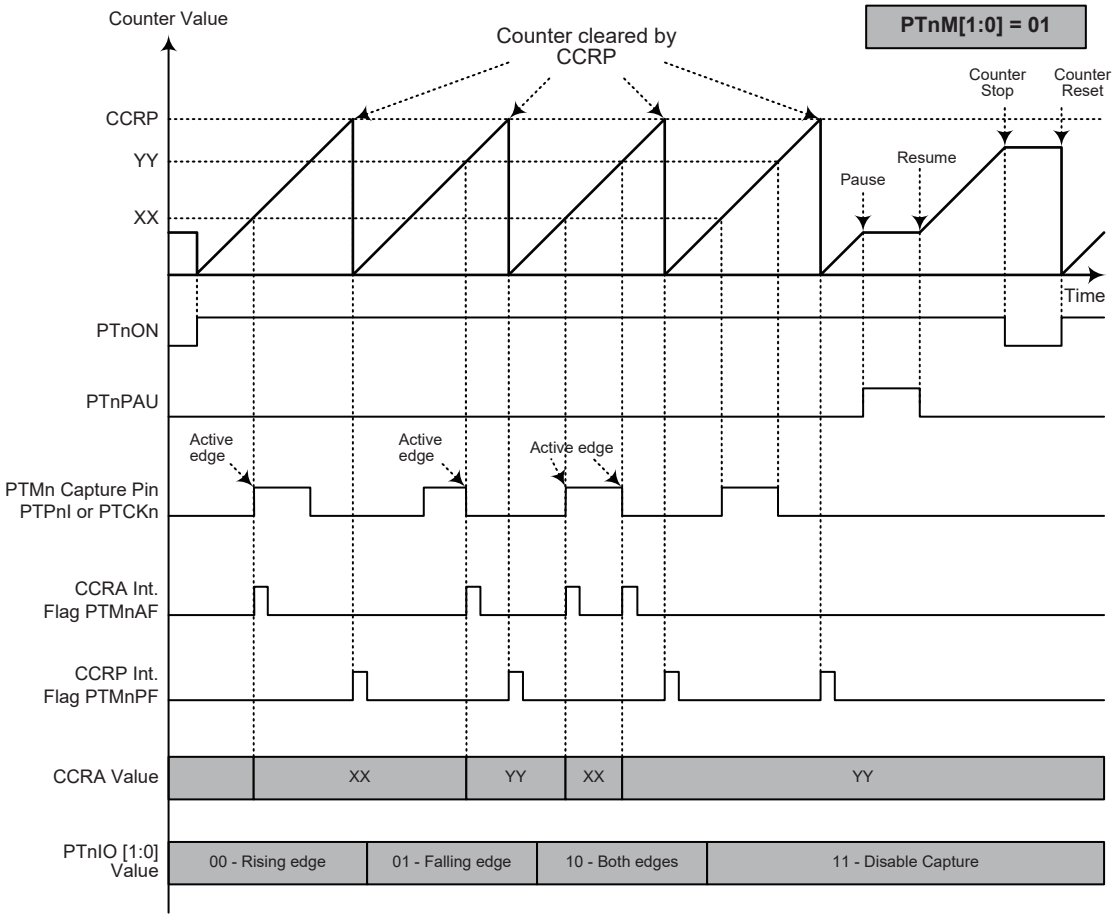
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse is triggered by the PTCKn pin or by setting the PTnON bit high
4. A PTCKn pin active edge will automatically set the PTnON bit high
5. In the Single Pulse Output Mode, PTnIO[1:0] must be set to "11" and cannot be changed.

Capture Input Mode

To select this mode bits PTnM1 and PTnM0 in the PTMnC1 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 PTPnI or PTCKn pin which is selected using the PTnCAPTS bit in the PTMnC1 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The counter is started when the PTnON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the PTPnI or PTCKn pin the present value in the counter will be latched into the CCRA registers and a PTMn interrupt generated. Irrespective of what events occur on the PTPnI or PTCKn pin, the counter will continue to free run until the PTnON 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 PTMn 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 PTnIO1 and PTnIO0 bits can select the active trigger edge on the PTPnI or PTCKn pin to be a rising edge, falling edge or both edge types. If the PTnIO1 and PTnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPnI or PTCKn pin, however it must be noted that the counter will continue to run.

As the PTPnI or PTCKn pin is pin shared with other functions, care must be taken if the PTMn is in the Capture Input Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The PTnCCLR, PTnOC and PTnPOL bits are not used in this Mode.



Capture Input Mode (n=0~2)

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

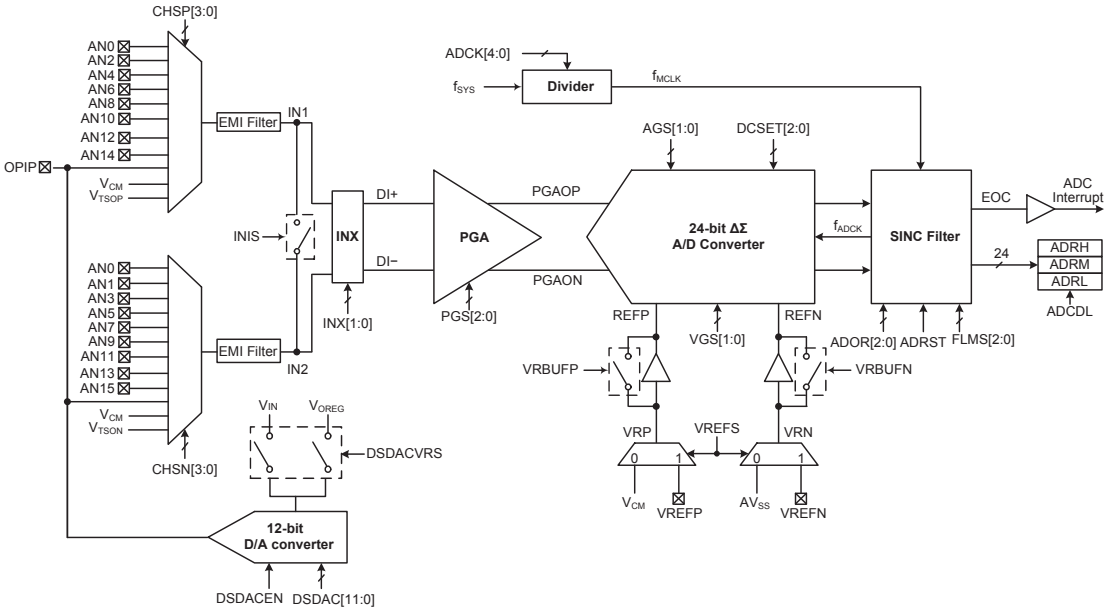
Analog to Digital Converter

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

These 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 sixteen single-ended A/D converter input channels or eight 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 SINC filter which can transform the converted one-bit data to 24 bits and store them into the specific data registers. Additionally, these devices also provide a temperature sensor to compensate the A/D converter deviation caused by the temperature. With high accuracy and performance, these devices are very suitable for differential output sensor applications such as weight measurement scales and other related products.

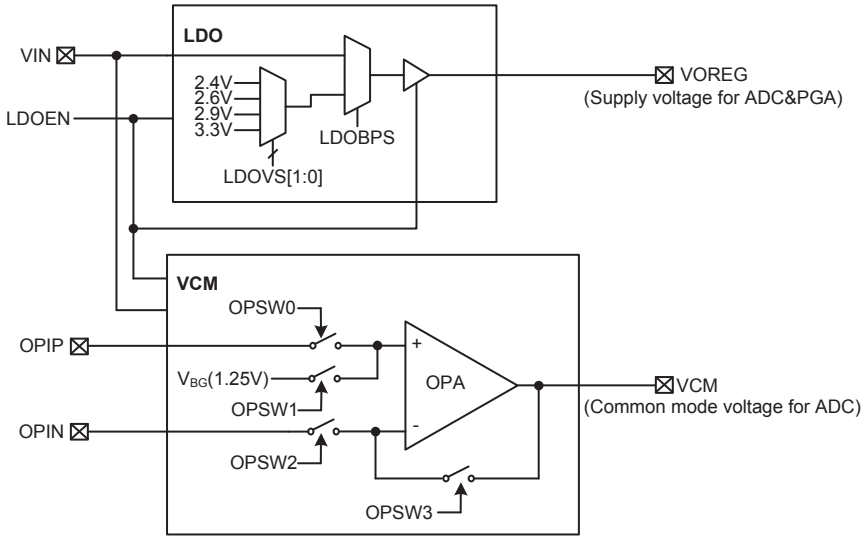


A/D Converter Structure

Internal Power Supply

These devices contain an LDO and VCM 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. The V_{CM} can be used as a reference voltage for the A/D converter module. 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 and VCM functions can be controlled by the LDOEN and ADOFF bits respectively and can be powered off to reduce overall power consumption. If the VCM is disabled, the VCM output pin is floating.



Internal Power Supply Block Diagram

Control Bits		Output Voltage		
ADOFF	**LDOEN**	**Bandgap**	**VOREG**	**VCM**
1	0	Off	Disable	Disable
1	1	On	Enable	Enable
0	0	On	Disable	Enable
0	1	On	Enable	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

• **DSOPC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	OPSW3	OPSW2	OPSW1	OPSW0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	0	1	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **OPSW3**: Switch control bit for OPA configuration
0: Off
1: On

Bit 2 **OPSW2**: Switch control bit for OPA configuration
0: Off
1: On

Bit 1 **OPSW1**: Switch control bit for OPA configuration
0: Off
1: On

Bit 0 **OPSW0**: Switch control bit for OPA configuration
0: Off
1: On

Note: This OPA can be used for signal amplification according to specific user requirements. With specific control registers, some OPA related applications can be more flexible and easier to be implemented. The initial state of OPA is a voltage follower for V_{BG}(1.25V).

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 $f_{SYS}/2/(ADCK[4:0]+1)$ using the ADCK bit field.

N: a constant divide factor equal to12 or 30 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 A/D converter 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 A/C 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 6 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
DSOPC	—	—	—	—	OPSW3	OPSW2	OPSW1	OPSW0
PGAC0	—	VGS1	VGS0	AGS1	AGS0	PGS2	PGS1	PGS0
PGAC1	—	INIS	INX1	INX0	DCSET2	DCSET1	DCSET0	—
PGACS	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
ADRL	D7	D6	D5	D4	D3	D2	D1	D0
ADRM	D15	D14	D13	D12	D11	D10	D9	D8
ADRH	D23	D22	D21	D20	D19	D18	D17	D16
ADCR0	ADRST	ADSLP	ADOFF	ADOR2	ADOR1	ADOR0	—	VREFS
ADCR1	FLMS2	FLMS1	FLMS0	VRBUFN	VRBUFP	ADCDL	EOC	—
ADCS	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
DSDAH	D11	D10	D9	D8	D7	D6	D5	D4
DSDAL	—	—	—	—	D3	D2	D1	D0
DSDACC	DSDACEN	DSDACVRS	—	—	—	—	—	—

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 input channels have to be determined by the CHSP3~CHSP0 and CHSN3~CHSN0 bits to 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

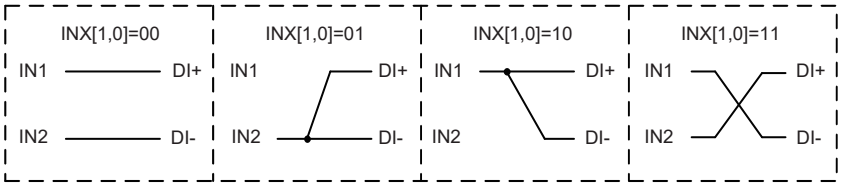
• **PGAC1 Register**

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: AN0
0001: AN1
0010: AN3
0011: AN5
0100: AN7
0101: AN9
0110: AN11
0111: AN13
1000:AN15
1001: OPIP
1010: Reserved
1011: V_{CM}
1100: Temperature sensor output – V_{TSON}
1101~1111: Reserved

These bits are used to select the negative input, IN2. If the IN2 input is selected as a single end input, the V_{CM} voltage must be selected as the positive input on IN1 for single end input applications. 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: AN4
0011: AN6
0100: AN8
0101: AN10
0110: AN12
0111: AN14
1000 : Reserved
1001: OPIP
1010: Reserved
1011: V_{CM}
1100: Temperature sensor output – V_{TSOP}
1101~1111: Reserved

These bits are used to select the positive input, IN1. If the IN1 input is selected as a single end input, the V_{CM} voltage must be selected as the negative input on IN2 for single end input applications. 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.

D/A Converter Registers – DSDAH, DSDAL, DSDACC

There are three registers related to the D/A converter control. Two data registers are used for D/A converter output control, one control register is used for D/A converter enable control and reference voltage selection.

• DSDAH Register

Bit	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	D7	D6	D5	D4
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 **D11~D4**: 12-bit D/A converter output control code bit 11 ~ bit 4

• DSDAL Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **D3~D0**: 12-bit D/A converter output control code bit 3 ~ bit 0

Note: Writing to this register only writes the low byte data to a shadow buffer. When writing to the DSDAH register, the shadow buffer data will also be copied into the DSDAL register.

• **DSDACC Register**

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

Bit 7 **DSDACEN**: D/A Converter enable or disable control bit
0: Disable
1: Enable

Bit 6 **DSDACVRS**: D/A Converter reference voltage selection
0: D/A converter reference voltage comes from VOREG
1: D/A converter reference voltage comes from VIN

Bit 5~0 Unimplemented, read as "0"

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 takes place, 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 selection of which reference source is used by the internal A/D converter, the A/D converter clock source, the A/D converter output 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	—	VREFS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	R/W
POR	0	0	1	0	0	0	—	0

Bit 7 **ADRST**: A/D converter software reset control
0: Disable
1: Enable
This bit is used to reset the A/D converter internal digital SINC filter. This bit is 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 set low 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 setting the ADOFF bit low. 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 for the PGA and internal Bandgap circuit to reduce overall power consumption and the V_{CM} start-up stable time.

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 Unimplemented, read as “0”

Bit 0 **VREFS**: A/D converter reference voltage pair selection
0: Internal reference voltage pair – V_{CM} & AV_{SS}
1: External reference voltage pair – V_{REFP} & V_{REFN}

• **ADCR1 Register**

Bit	7	6	5	4	3	2	1	0
Name	FLMS2	FLMS1	FLMS0	VRBUFN	VRBUFP	ADCDL	EOC	—
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 ~ 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 rate 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 **VRBUFN**: A/D converter negative reference voltage input (VRN) buffer control
0: Disable input buffer and enable bypass function
1: Enable input buffer and disable bypass function

Bit 3 **VRBUFP**: A/D converter positive reference voltage input (VRP) buffer control
0: Disable input buffer and enable bypass function
1: Enable input buffer and disable bypass function

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} divided ratio selection
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, VCM off, PGA off, ADC off, Temperature sensor off, VRN/VRP buffer off, SINC filter off
1	1	x	x	Power down mode	Bandgap on, LDO on, VCM on, PGA off, ADC off, Temperature sensor off, VRN/VRP buffer off, SINC filter off
0	0	1	x	Sleep mode (External voltage must be supplied on LDO output pin)	Bandgap on, LDO off, V_{CM} on, PGA on, ADC off, Temperature sensor off, VRN/VRP buffer off, SINC filter on
0	0	0	0	Normal mode (External voltage must be supplied on LDO output pin)	Bandgap on, LDO off, V_{CM} on, PGA on, ADC on, Temperature sensor on/off(2), VRN/VRP buffer on/off(3), SINC filter on
0	0	0	1	Reset mode (External voltage must be supplied on LDO output pin)	Bandgap on, LDO off, V_{CM} on, PGA on, ADC on, Temperature sensor on/off(2), VRN/VRP buffer on/off(3), SINC filter Reset
1	0	1	x	Sleep mode	Bandgap on, LDO on, V_{CM} on, PGA on, ADC off, Temperature sensor off, VRN/VRP buffer off, SINC filter on
1	0	0	0	Normal mode	Bandgap on, LDO on, V_{CM} on, PGA on, ADC on, Temperature sensor on/off(2), VRN/VRP buffer on/off(3), SINC filter on
1	0	0	1	Reset mode	Bandgap on, LDO on, V_{CM} on, PGA on, ADC on, Temperature sensor on/off(2), VRN/VRP buffer on/off(3), SINC filter Reset

Note: 1. The V_{CM} on/off function is controlled directly by the bandgap on/off condition.
2. The Temperature Sensor can be switched on or off by configuring the CHSN[3:0] or CHSP[3:0] bits.
3. The VRN buffer can be switched on or off by configuring the VRBUFN bit while the VRP buffer can be switched on or off by configuring the VRBUFP bit
4. "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 "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.

The differential reference voltage supply to the A/D Converter can be supplied from either the internal power supply, V_{CM} and AV_{SS}, or from an external reference source supplied on pins, VREFP and VREFN. The desired selection is made using the VREFS bit in the ADCR0 register.

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, VCM for PGA and ADC.
- Step 2
  Select the PGA, ADC, reference voltage gains by PGAC0 register
- Step 3
  Select the PGA settings for input connection and input offset by PGAC1 register
- Step 4
  Select the required A/D conversion clock source by correctly programming bits ADCK4~ADCK0 in the ADCS register.
- Step 5
  Select output data rate by configuring the ADOR[2:0] bits in the ADCR0 register and FLMS[2:0] bits in the ADCR1 register.
- Step 6
  Select which channel is to be connected to the internal PGA by correctly programming the CHSP3~CHSP0 and CHSN3~CHSN0 bits which are also 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 to high in the ADCR0 register and then clearing this bit to zero to release the reset status.
- Step 9
  If the interrupts are 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 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

These 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 V_{CM} or the differential reference input voltage, ΔVR_I, selected by the VREFS bit in ADCR0 register, this gives a single bit analog input value of ΔVR_I divided by 8388608.

1 LSB=ΔVR_I / 8388608

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

ΔSI_I=(PGAGN × ADGN × ΔDI±) + DCSET

ΔVR_I=VREFGN × ΔVR±

ADC_Conversion_Data=(ΔSI_I / ΔVR_I) × 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

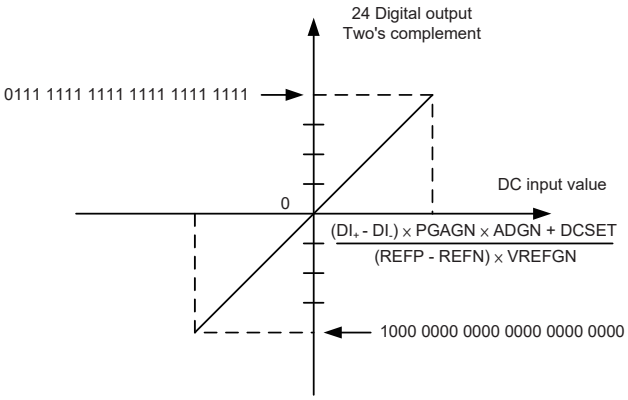
Due to the digital system design of the Delta Sigma A/D Converter, the maximum A/D converted value is 8388607 and the minimum value is -8388608. Therefore, there is a middle value of 0. The ADC_Conversion_Data equation illustrates this range of converted data variation.

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

$$\text{Input Voltage} = \frac{(\text{Converted_data}) \times \text{LSB} - \text{DCSET}}{\text{PAG} \times \text{ADGN}}$$

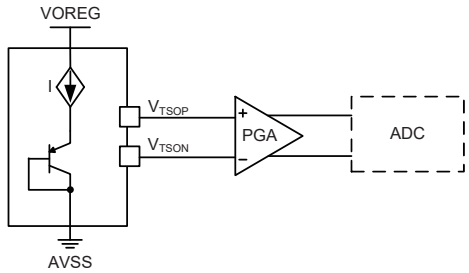
If the MSB = 1 – Negative Converted data

$$\text{Input voltage} = \frac{(\text{Two's_complement_of_Converted_data}) \times \text{LSB} - \text{DCSET}}{\text{PAG} \times \text{ADGN}}$$

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

Temperature Sensor

An internal temperature sensor is integrated within the device 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

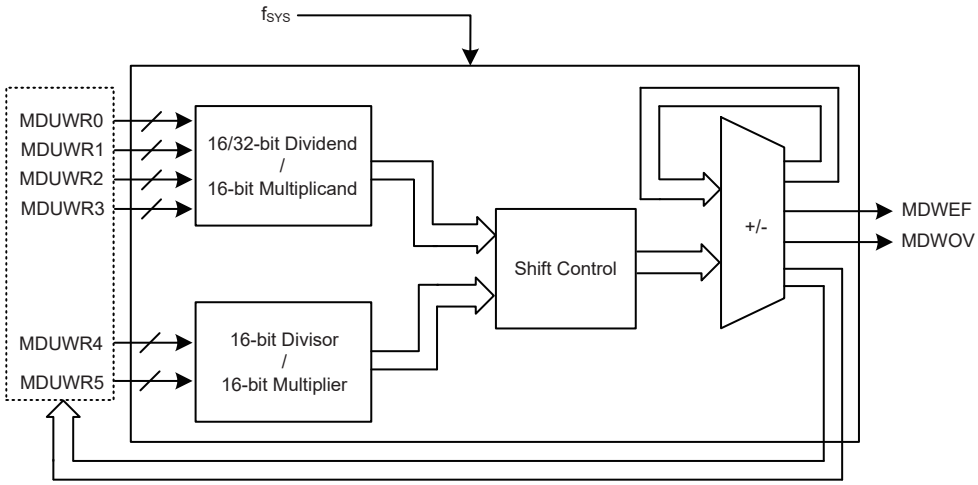
A/D Conversion Programming Example

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

```
#include BH67f5250.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, 083H                ; Power control for PGA, ADC
     mov PWRC, a                ; PWRC=10000011, LDO enable, VCM enable,
                                ; LDO Bypass disable, LDO output voltage: 3.3V
     mov a, 000H
     mov PGAC0, a               ; PGA gain=1, ADC gain=1, VREF gain=1
     mov a, 000H
     mov PGAC1, a               ; INIS, INX, DCSET in default value
     clr VRBUFP                 ; disable buffer for VREF+
     clr VRBUFN                 ; disable buffer for VREF-
     set VREFS                  ; for using external reference
     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
```

16-bit Multiplication Division Unit – MDU

These devices have a 16-bit Multiplication Division Unit, MDU, which integrates a 16-bit unsigned multiplier and a 32-bit/16-bit divider. The MDU, in replacing the software multiplication and division operations, can therefore save large amounts of computing time as well as the Program and Data Memory space. It also reduces the overall microcontroller loading and results in the overall system performance improvements.



16-Bit MDU Block Diagram

MDU Registers

The multiplication and division operations are implemented in a specific way, a specific write access sequence of a series of MDU data registers. The status register, MDUWCTRL, provides the indications for the MDU operation. The data register each is used to store the data regarded as the different operand corresponding to different MDU operations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
MDUWR0	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR1	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR2	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR3	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR4	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR5	D7	D6	D5	D4	D3	D2	D1	D0
MDUWCTRL	MDWEF	MDWOV	—	—	—	—	—	—

MDU Register List

- **MDUWRn Register (n=0~5)**

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

"x": unknown

Bit 7~0 **D7~D0**: 16-bit MDU data register n

• **MDUWCTRL Register**

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

Bit 7 **MDWEF**: 16-bit MDU error flag
0: Normal
1: Abnormal
This bit will be set high if the data register MDUWRn is written or read as the MDU operation is executing. This bit should be cleared to zero by reading the MDUWCTRL register if it is equal to 1 and the MDU operation is completed.

Bit 6 **MDWOV**: 16-bit MDU overflow flag
0: No overflow occurs
1: Multiplication product > FFFFH or Divisor=0
When an operation is completed, this bit will be updated by hardware to a new value corresponding to the current operation situation.

Bit 5~0 Unimplemented, read as "0"

MDU Operation

For this MDU the multiplication or division operation is carried out in a specific way and is determined by the write access sequence of the six MDU data registers, MDUWR0~MDUWR5. The low byte data, regardless of the dividend, multiplicand, divisor or multiplier, must first be written into the corresponding MDU data register followed by the high byte data. All MDU operations will be executed after the MDUWR5 register is write-accessed together with the correct specific write access sequence of the MDUWRn. Note that it is not necessary to consecutively write data into the MDU data registers but must be in a correct write access sequence. Therefore, a non-write MDUWRn instruction or an interrupt, etc., can be inserted into the correct write access sequence without destroying the write operation. The relationship between the write access sequence and the MDU operation is shown in the following.

- 32-bit/16-bit division operation: Write data sequentially into the six MDU data registers from MDUWR0 to MDUWR5.
- 16-bit/16-bit division operation: Write data sequentially into the specific four MDU data registers in a sequence of MDUWR0, MDUWR1, MDUWR4 and MDUWR5 with no write access to MDUWR2 and MDUWR3.
- 16-bit×16-bit multiplication operation: Write data sequentially into the specific four MDU data registers in a sequence of MDUWR0, MDUWR4, MDUWR1 and MDUWR5 with no write access to MDUWR2 and MDUWR3.

After the specific write access sequence is determined, the MDU will start to perform the corresponding operation. The calculation time necessary for these MDU operations are different. During the calculation time any read/write access to the six MDU data registers is forbidden. After the completion of each operation, it is necessary to check the operation status in the MDUWCTRL register to make sure that whether the operation is correct or not. Then the operation result can be read out from the corresponding MDU data registers in a specific read access sequence if the operation is correctly finished. The necessary calculation time for different MDU operations is listed in the following.

- 32-bit/16-bit division operation: $17\times t_{SYS}$.
- 16-bit/16-bit division operation: $9\times t_{SYS}$.
- 16-bit×16-bit multiplication operation: $11\times t_{SYS}$.

The operation results will be stored in the corresponding MDU data registers and should be read out from the MDU data registers in a specific read access sequence after the operation is completed. Noe that it is not necessary to consecutively read data out from the MDU data registers but must be in a correct read access sequence. Therefore, a non-read MDUWRn instruction or an interrupt, etc., can be inserted into the correct read access sequence without destroying the read operation. The relationship between the operation result read access sequence and the MDU operation is shown in the following.

- 32-bit/16-bit division operation: Read the quotient from MDUWR0 to MDUWR3 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit/16-bit division operation: Read the quotient from MDUWR0 and MDUWR1 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit×16-bit multiplication operation: Read the product sequentially from MDUWR0 to MDUWR3.

The overall important points for the MDU read/write access sequence and calculation time are summarized in the following table. Note that the device should not enter the IDLE or SLEEP mode until the MDU operation is totally completed, otherwise the MDU operation will fail.

Items \ Operations	32-bit / 16-bit Division	16-bit / 16-bit Division	16-bit × 16-bit Multiplication
Write Sequence First write ↓ ↓ ↓ ↓ Last write	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Dividend Byte 2 written to MDUWR2 Dividend Byte 3 written to MDUWR3 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Multiplicand Byte 0 written to MDUWR0 Multiplier Byte 0 written to MDUWR4 Multiplicand Byte 1 written to MDUWR1 Multiplier Byte 1 written to MDUWR5
Calculation Time	17 × t_{SYS}	9 × t_{SYS}	11 × t_{SYS}
Read Sequence First read ↓ ↓ ↓ ↓ Last read	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Quotient Byte 2 read from MDUWR2 Quotient Byte 3 read from MDUWR3 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Product Byte 0 read from MDUWR0 Product Byte 1 read from MDUWR1 Product Byte 2 read from MDUWR2 Product Byte 3 read from MDUWR3

MDU Operations Summary

LCD Driver

For large volume applications, which incorporate an LCD in their design, the use of a custom display rather than a more expensive character based display reduces costs significantly. However, the corresponding COM and SEG signals required, which vary in both amplitude and time, to drive such a custom display require many special considerations for proper LCD operation to occur. These devices contain an LCD Driver function, which with their internal LCD signal generating circuitry and various options will automatically generate these time and amplitude varying signals to provide a means of direct driving and easy interfacing to a range of custom LCDs.

These devices include a wide range of options to enable LCD displays of various types to be driven. The tables show the range of options available across the devices range.

Driver No.	Duty	Bias Level	Bias Type	Waveform Type
28×4	1/4	1/3	R or C	A or B
26×6	1/6	1/3	R or C	A or B
24×8	1/8	1/3	R	A or B
24×8	1/8	1/4	R	A or B

LCD Driver Output Selection – BH67F5250

Driver No.		Duty	Bias Level	Bias Type	Waveform Type
80LQFP	64LQFP				
42×4	28×4	1/4	1/3	R or C	A or B
40×6	26×6	1/6	1/3	R or C	A or B
38×8	24×8	1/8	1/3	R	A or B
38×8	24×8	1/8	1/4	R	A or B

LCD Driver Output Selection – BH67F5260/BH67F5270

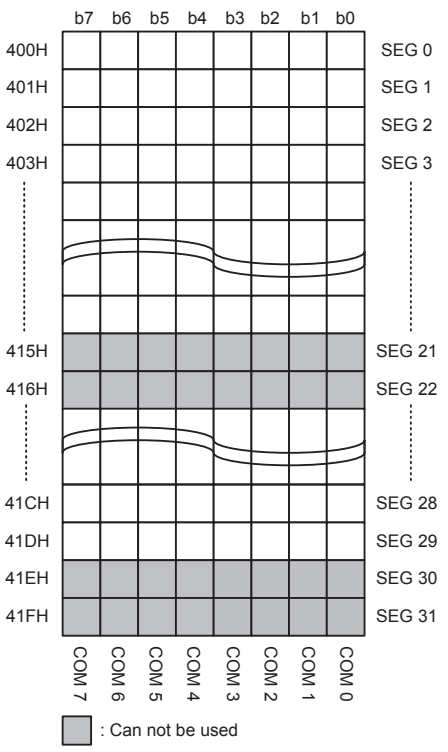
LCD Display Memory

An area of Data Memory is especially reserved for use for the LCD display data. This data area is known as the LCD Memory. Any data written here will be automatically read by the internal display driver circuits, which will in turn automatically generate the necessary LCD driving signals. Therefore any data written into this Memory will be immediately reflected into the actual display connected to the microcontroller.

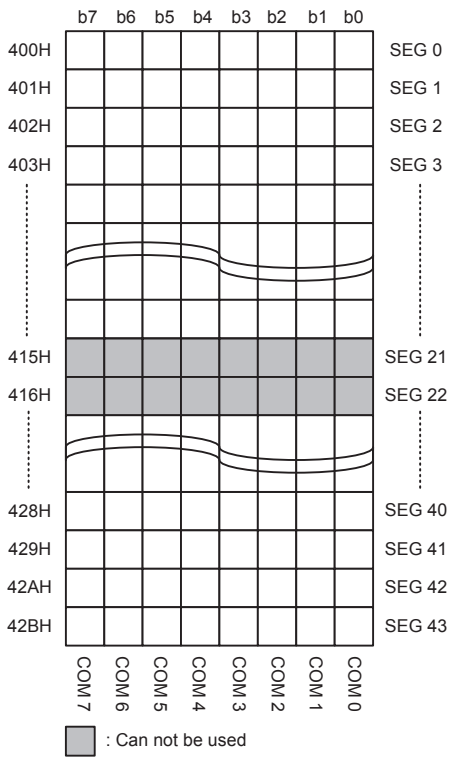
These devices provide an area of embedded data memory for the LCD display. This area is located at 00H to 1FH or 00H to 2BH in Sector 4 of the Data Memory respectively. The LCD display memory can be read and written to by indirect addressing mode using MP1L/MP1H and MP2L/MP2H, or by direct addressing mode using the corresponding extended instructions. If using the indirect addressing to access the Display Memory therefore requires first that Sector 4 is selected by writing a value of 04H to MP1H or MP2H. After this, the memory can then be accessed by using indirect addressing through the use of MP1L or MP2L. With Sector 4 selected, then using MP1L/MP2L to read or write to the memory area, from 00H to 1FH or 00H to 2BH, will result in operations to the LCD memory.

When data is written into the display data area, it is automatically read by the LCD driver which then generates the corresponding LCD driving signals. To turn the display on or off, a “1” or a “0” is written to the corresponding bit of the display memory, respectively. The figure illustrates the mapping between the display memory and LCD pattern for the device.

The unimplemented LCD RAM bits cannot be used as general purpose RAM for application. For example, if the LCD duty is selected as 1/4 duty (4COM), the COM b4~b7 will be read as 0 only.



LCD Memory Map – BH67F5250



LCD Memory Map – BH67F5260/BH67F5270

LCD Clock Source

The LCD clock source is the internal clock signal, f_{SUB}, divided by 8, using an internal divider circuit. The f_{SUB} internal clock is supplied by either the LIRC or LXT oscillator, the choice of which is determined by the FSS bit in the SCC register. For proper LCD operation, this arrangement is provided to generate an ideal LCD clock source frequency of 4kHz.

f_{SUB} Clock Source	LCD Clock Frequency
LIRC	4kHz
LXT	4kHz

LCD Clock Source

LCD Registers

Control Registers in the Data Memory, are used to control the various setup features of the LCD Driver. There are several control registers for the LCD function, LCDCP, LCDC0 and LCDC2.

The LCDPR bit in the LCDCP register is used to select the PLCD pin or the internal charge pump regulator to supply the power for the R type LCD COMs and SEGs pins. Bits CPVS1 and CPVS0 in the same register are used to select an appropriate charge pump output voltage level for the R type LCD.

The TYPE bit in the LCDC0 register is used to select whether Type A or Type B LCD control signals are used. The RCT bit in the LCDC0 register is used to select whether R type or C type LCD drive bias. Bits LCDP1 and LCDP0 in the LCDC0 register are used to select the power source to supply the C type LCD panel with the correct bias voltages. Bits LCDIS1 and LCDIS0 in the LCDC0 register are used to select the internal bias current to supply the R type LCD panel with the correct bias voltages. A choice to best match the LCD panel used in the application can be selected

also to minimise bias current. The LCDEN bit in the LCDC0 register, which provides the overall LCD enable/disable function, will only be effective when this device is in the FAST, SLOW or IDLE Mode. If this device is in the SLEEP Mode then the display will always be disabled.

The LCDC2 register is used for LCD duty and bias selection.

Register Name	Bit							
	7	6	5	4	3	2	1	0
LCDC0	TYPE	RCT	LCDP1	LCDP0	—	LCDIS1	LCDIS0	LCDEN
LCDCP	—	—	—	—	LCDPR	—	CPVS1	CPVS0
LCDC2	LCDPCK2	LCDPCK1	LCDPCK0	—	—	DTYC1	DTYC0	BIAS

LCD Register List

• LCDC0 Register

Bit	7	6	5	4	3	2	1	0
Name	TYPE	RCT	LCDP1	LCDP0	—	LCDIS1	LCDIS0	LCDEN
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 **TYPE**: LCD Waveform Type Selection
0: Type A
1: Type B

Bit 6 **RCT**: R or C LCD Type Selection
0: R type
1: C type
When the C type is selected, the LCDPR bit in the LCDCP register should be fixed at 0.

Bit 5~4 **LCDP1~LCDP0**: LCD power source selection for C type LCD
00: From external pin PLCD, V1 or V2
01: From internal reference voltage V_{REFIN} supplied to V_C
10: From internal voltage V_{DD} supplied to V_B
11: From internal voltage V_{DD} supplied to V_A
The V_{REFIN} is an internal reference voltage with an approximate level of 1.08V.

Bit 3 Unimplemented, read as "0"

Bit 2~1 **LCDIS1~LCDIS0**: LCD Bias Current Selection for R type LCD (V_A=V_{PLCD}=V_{DD}, 1/3 bias)
00: 25μA
01: 50μA
10: 100μA
11: 200μA
When using the C type LCD, these bits should be fixed at 00.

Bit 0 **LCDEN**: LCD Enable Control
0: Disable
1: Enable
In the FAST, SLOW or IDLE mode, the LCD on/off function can be controlled by this bit. In the SLEEP mode, the LCD is always off.

• LCDCP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	LCDPR	—	CPVS1	CPVS0
R/W	—	—	—	—	R/W	—	R/W	R/W
POR	—	—	—	—	0	—	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **LCDPR**: LCD Power selection for R type
0: R PLCD pin
1: R type internal charge pump
This bit is only available for R type LCD applications. When the LCDPR bit is cleared to zero, the R type LCD power will be derived from the PLCD pin and the internal charge pump circuit will be disabled. This internal charge pump will also be disabled when the C type LCD driver is selected by setting the RCT bit to 1 or the LCD driver is disabled by clearing the LCDEN bit to 0.

Bit 2 Unimplemented, read as "0"

Bit 1~0 **CPVS1~CPVS0**: Charge pump output voltage selection for R type
00: 3.3V
01: 3.0V
10: 2.7V
11: 4.5V

• **LCDC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	LCDPCK2	LCDPCK1	LCDPCK0	—	—	DTYC1	DTYC0	BIAS
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7~5 **LCDPCK2~LCDPCK0**: C type LCD Pump Clock divider
000: 250Hz (f_{SUB}/128)
001: 500Hz (f_{SUB}/64)
010: 1kHz (f_{SUB}/32)
011: 2kHz (f_{SUB}/16)
100: 4kHz (f_{SUB}/8)
101: 8kHz (f_{SUB}/4)
110: 16kHz (f_{SUB}/2)
111: 16kHz (f_{SUB}/2)

Bit 4~3 Unimplemented, read as "0"

Bit 2~1 **DTYC1~DTYC0**: LCD duty selection
00: 1/4 Duty (COM0~COM3)
01: 1/6 Duty (COM0~COM5)
10: 1/8 Duty (COM0~COM7, for R type only)
11: undefined
The unused COMn pins are allowed to be configured as normal I/O or other pin-shared functions.

Bit 0 **BIAS**: LCD bias selection
0: 1/3 bias
1: 1/4 bias, for R type only

LCD Voltage Source and Biasing

The time and amplitude varying signals generated by the LCD Driver function require the generation of several voltage levels for their operation. The device can have either R type or C type biasing selected via a software control bit RCT. Selecting the C type biasing will enable C type internal charge pump circuitry.

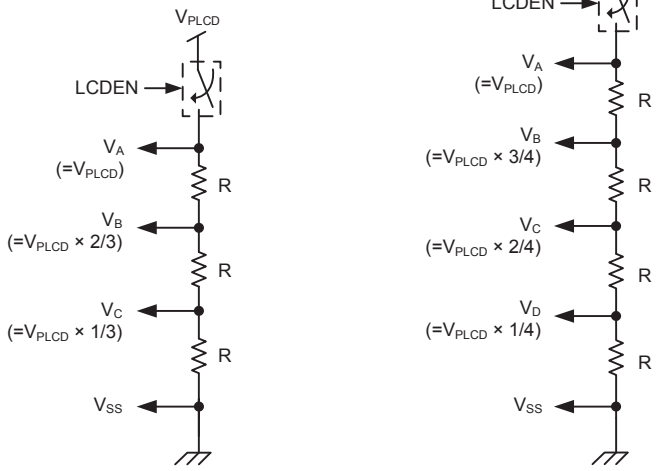
R Type Biasing

For R type biasing the LCD voltage source, the PLCD voltage can be supplied by the PLCD pin or internal charge pump regulator, selected by the LCDPR bit in the LCDCP register, to generate the internal biasing voltages. The source on the PLCD pin could be the microcontroller power supply or some other voltage source. There are four kinds of the internal charge pump voltage output,

managed by the CPVS[1:0] bits in the LCDCP register.

For the R type 1/3 bias scheme, four voltage levels V_{SS}, V_A, V_B and V_C are utilised. The voltage V_A is equal to V_{PLCD}. The voltage V_B is equal to $V_{PLCD}\times2/3$ while the voltage V_C is equal to $V_{PLCD}\times1/3$.

For the R type 1/4 bias scheme, five voltage levels V_{SS}, V_A, V_B, V_C and V_D are utilised. The voltage V_A is equal to V_{PLCD}. The voltage V_B is equal to $V_A\times3/4$, the voltage V_C is equal to $V_A\times2/4$ and the voltage V_D is equal to $V_A\times1/4$.



R Type 1/3 Bias **R Type 1/4 Bias**

Note: 1. The DC path will be switched off when the LCD is disabled.
2. When LCDPR=1, the PLCD pin should externally connect a 4.7μF capacitor; when LCDPR=0, the PLCD pin does not require an external capacitor.

R Type Bias Voltage Generation

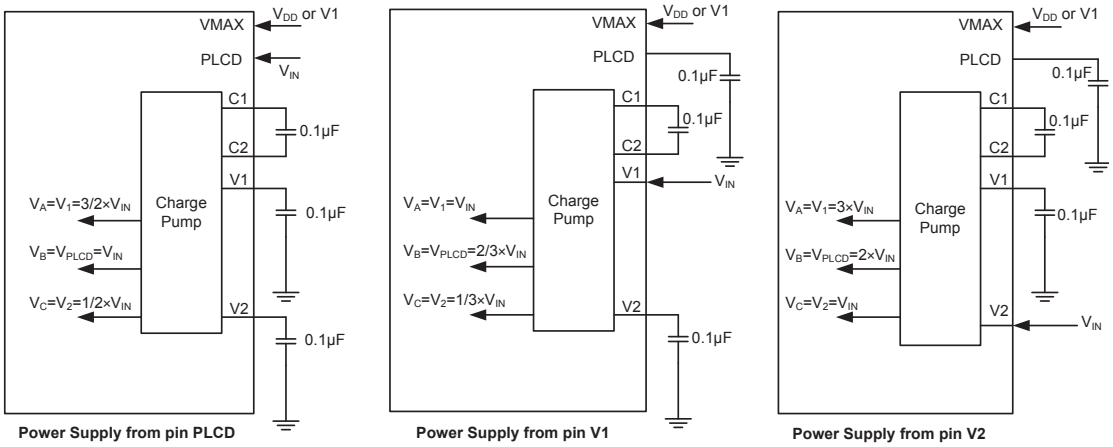
Different values of internal bias current can be selected using the LCDIS1~LCDIS0 bits in the LCDC0 register. The VMAX pin should be connected to the PLCD or VDD pin which provides the maximum voltage.

C Type Biasing

For C type biasing the LCD voltage source can be supplied on the external pin PLCD, V1 or V2 or derived from the internal voltage source to generate the required biasing voltages. The C type bias voltage source is selected using the LCDP1 and LCDP0 bits in the LCDC0 register.

When the LCD voltage source is from the PLCD pin, the C type biasing scheme uses an internal charge pump circuit, which can generate voltages higher than what is supplied on PLCD. This feature is useful in applications where the microcontroller supply voltage is less than the supply voltage required by the LCD. The charge pump clock divider is selected using the LCDPCK2~LCDPCK0 bits in the LCDC2 register. An additional charge pump capacitor must also be connected between pins C1 and C2 to generate the necessary voltage levels.

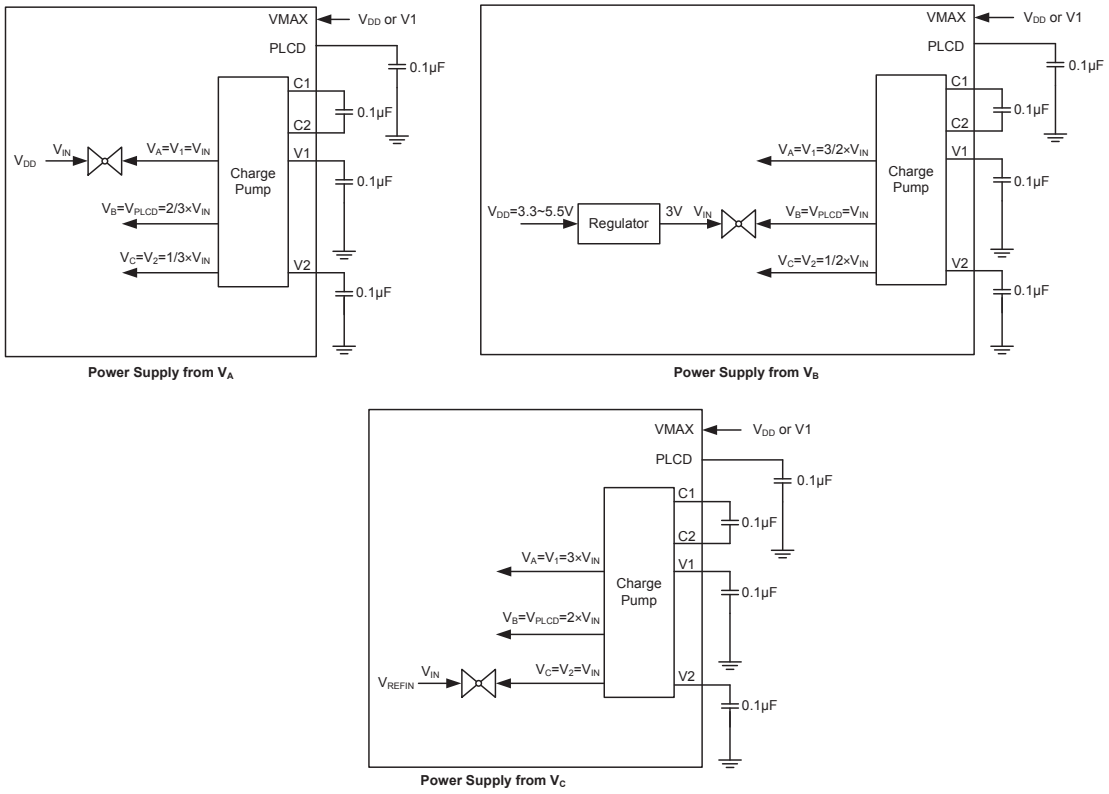
For C type 1/3 bias external power supply scheme, the LCD power can be supplied on PLCD, V1 or V2 pin. However, the LCD power is internally supplied on V_A, V_B or V_C for C type 1/3 bias internal power supply scheme. Four internally generated voltage levels V_{SS}, V_A, V_B and V_C are utilised. These bias voltages have different levels depending upon different LCD power supply schemes.



Power Supply from pin PLCD **Power Supply from pin V1** **Power Supply from pin V2**

Note: The pin VMAX must be connected to the maximum voltage to prevent from the pad leakage.

C Type Bias External Power Supply Configuration – 1/3 Bias



Power Supply from V_A **Power Supply from V_B**

Power Supply from V_C

Note: The pin VMAX must be connected to the maximum voltage to prevent from the pad leakage.

C Type Bias Internal Power Supply Configuration – 1/3 Bias

LCD Power Supply		V_A Voltage	V_B Voltage	V_C Voltage
External Power Supply	V_{IN} on V1	V_{IN}	2/3 × V_{IN}	1/3 × V_{IN}
	V_{IN} on PLCD	3/2 × V_{IN}	V_{IN}	1/2 × V_{IN}
	V_{IN} on V2	3 × V_{IN}	2 × V_{IN}	V_{IN}
Internal Power Supply	V_{DD} on V_A	V_{DD}	2/3 × V_{DD}	1/3 × V_{DD}
	V_{DD} on V_B	3/2 × V_{DD}	V_{DD}	1/2 × V_{DD}
	V_{REFIN} on V_C	3 × V_{REFIN}	2 × V_{REFIN}	V_{REFIN}

C Type Bias Power Supply Scheme

The connection to the VMAX pin depends upon the LCD power supply scheme. It is extremely important to ensure that these charge pump generated internal voltages do not exceed the maximum V_{DD} voltage of 5.5V.

Condition	VMAX Connection
V_{DD} > V_{IN} × 1.5	Connect VMAX to VDD
Otherwise	Connect VMAX to V1

C Type Bias VMAX Pin Connection

LCD Reset Function

The LCD has an internal reset function that is an OR function of the inverted LCDEN bit in the LCDC0 register and the SLEEP function. Clearing the LCDEN bit to zero will also reset the LCD function. The LCD function will be reset after the device enters the SLEEP mode even if the LCDEN bit is set high to enable the LCD driver function.

When the LCDEN bit is set high to enable the LCD driver and then an MCU reset occurs, the LCD driver will be reset and the COM and SEG outputs will be in a floating state during the MCU reset duration. The reset operation will take a time of t_{RSTD}+t_{SST}. Refer to the System Start Up Time Characteristics for t_{RSTD} and t_{SST} details.

MCU Reset	SLEEP Mode	LCDEN	LCD Reset	COM & SEG Voltage Level
No	Off	1	No	Normal Operation
No	Off	0	Yes	Low
No	On	x	Yes	Low
Yes	x	x	Yes	Floating

Note: 1. The Watchdog time-out reset in the IDLE or SLEEP Mode is excluded from the MCU Reset conditions.
2. “x”: Don’t care

LCD Reset Status

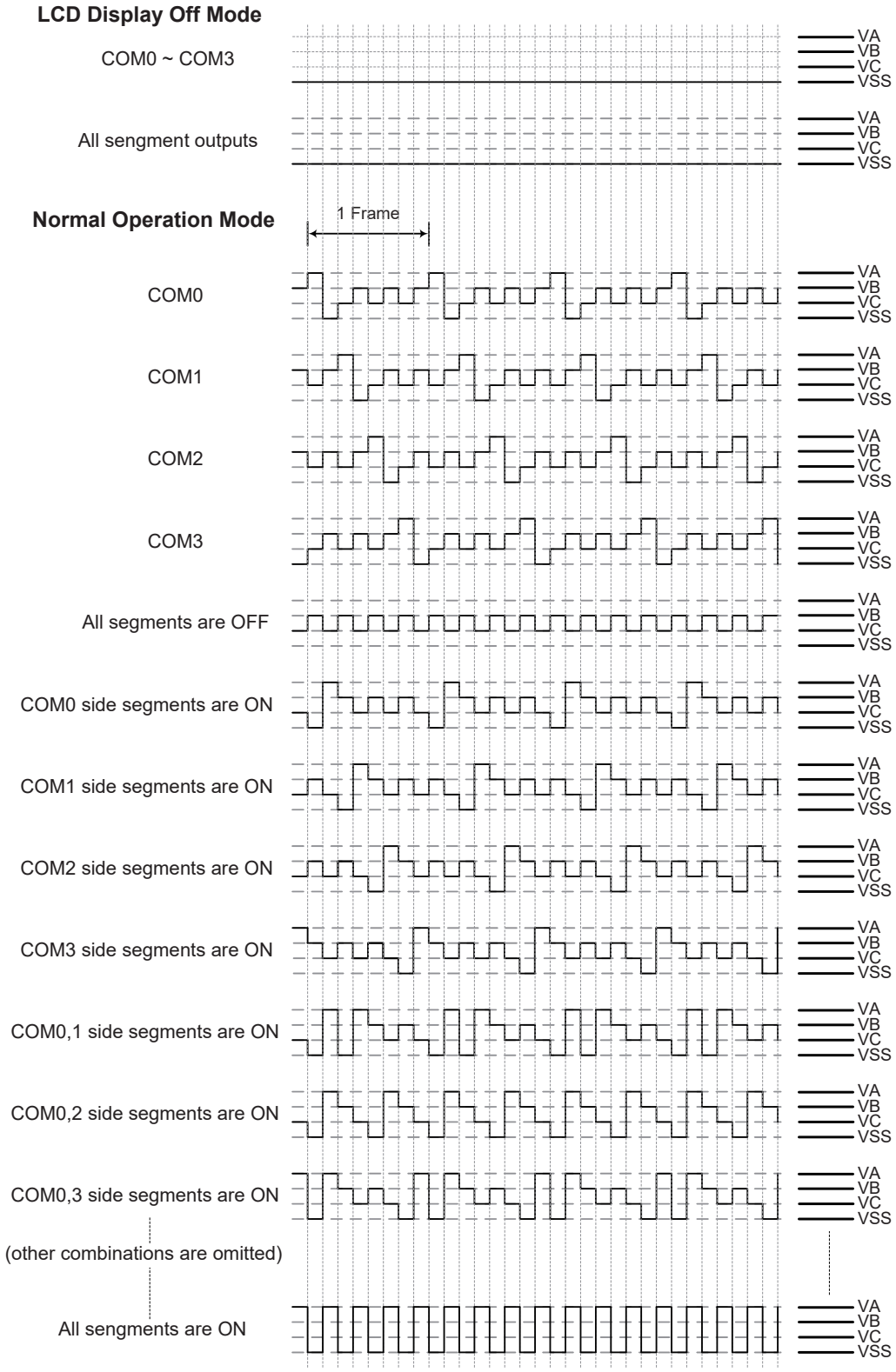
LCD Driver Output

The number of COM and SEG outputs supplied by the LCD driver, as well as its biasing and wave type selections, are dependent upon how the LCD control bits are programmed. The Bias Type, whether C or R type is also selected by a software control bit.

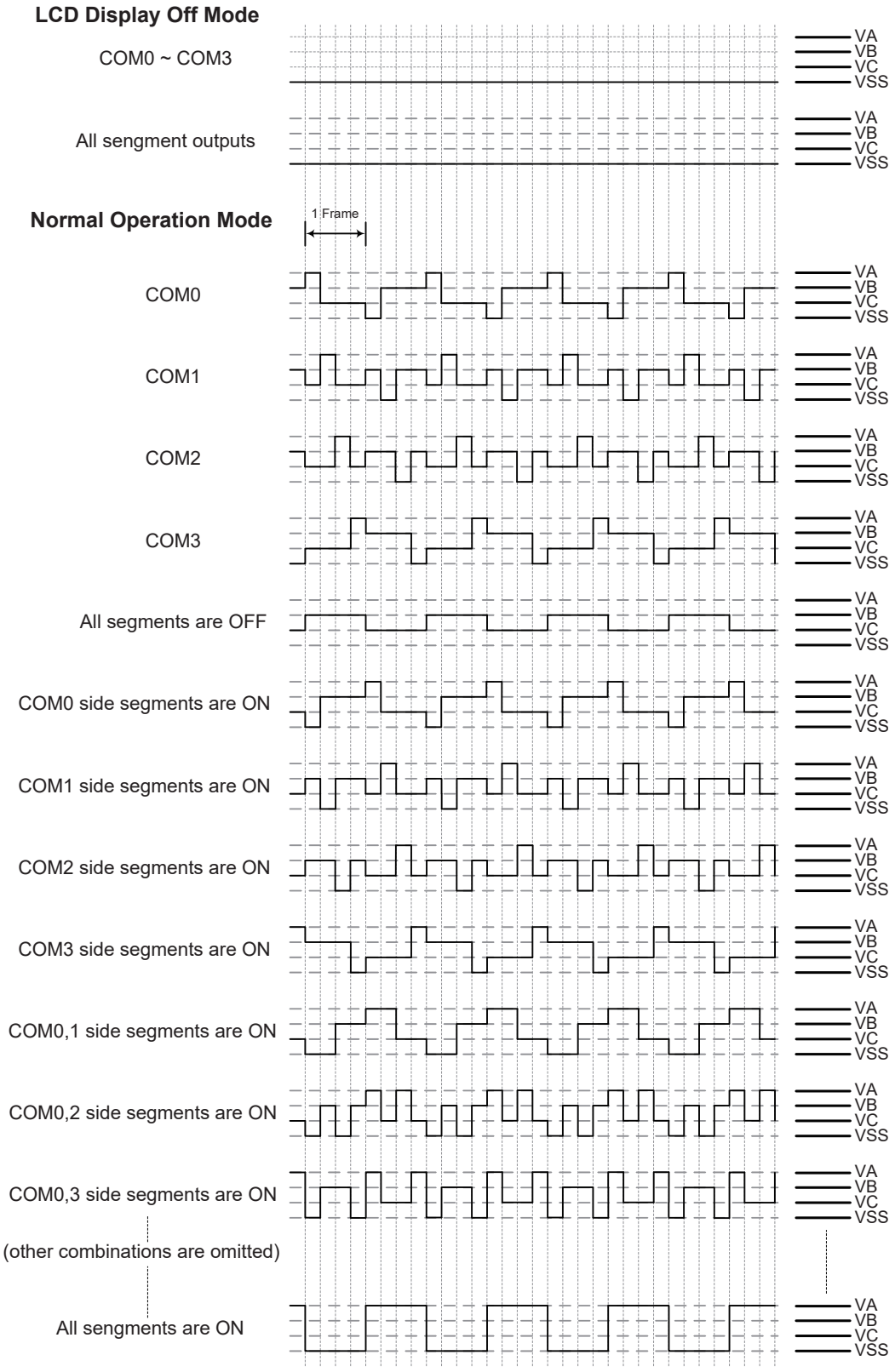
The nature of Liquid Crystal Displays require that only AC voltages can be applied to their pixels as the application of DC voltages to LCD pixels may cause permanent damage. For this reason the relative contrast of an LCD display is controlled by the actual RMS voltage applied to each pixel, which is equal to the RMS value of the voltage on the COM pin minus the voltage applied to the SEG pin. This differential RMS voltage must be greater than the LCD saturation voltage for the pixel to be on and less than the threshold voltage for the pixel to be off.

The requirement to limit the DC voltage to zero and to control as many pixels as possible with a minimum number of connections requires that both a time and amplitude signal is generated and applied to the application LCD. These time and amplitude varying signals are automatically generated by the LCD driver circuits in the microcontroller. What is known as the duty determines the number of common lines used, which are also known as backplanes or COMs. For example, the duty is 1/4 and equates to a COM number of 4, therefore defines the number of time divisions within each LCD signal frame. Two types of signal generation are also provided, known as Type A and Type B, the required type is selected via the TYPE bit in the LCDC0 register. Type B offers lower frequency signals, however lower frequencies may introduce flickering and influence display clarity.

R & C Type, 4-COM, 1/3 Bias

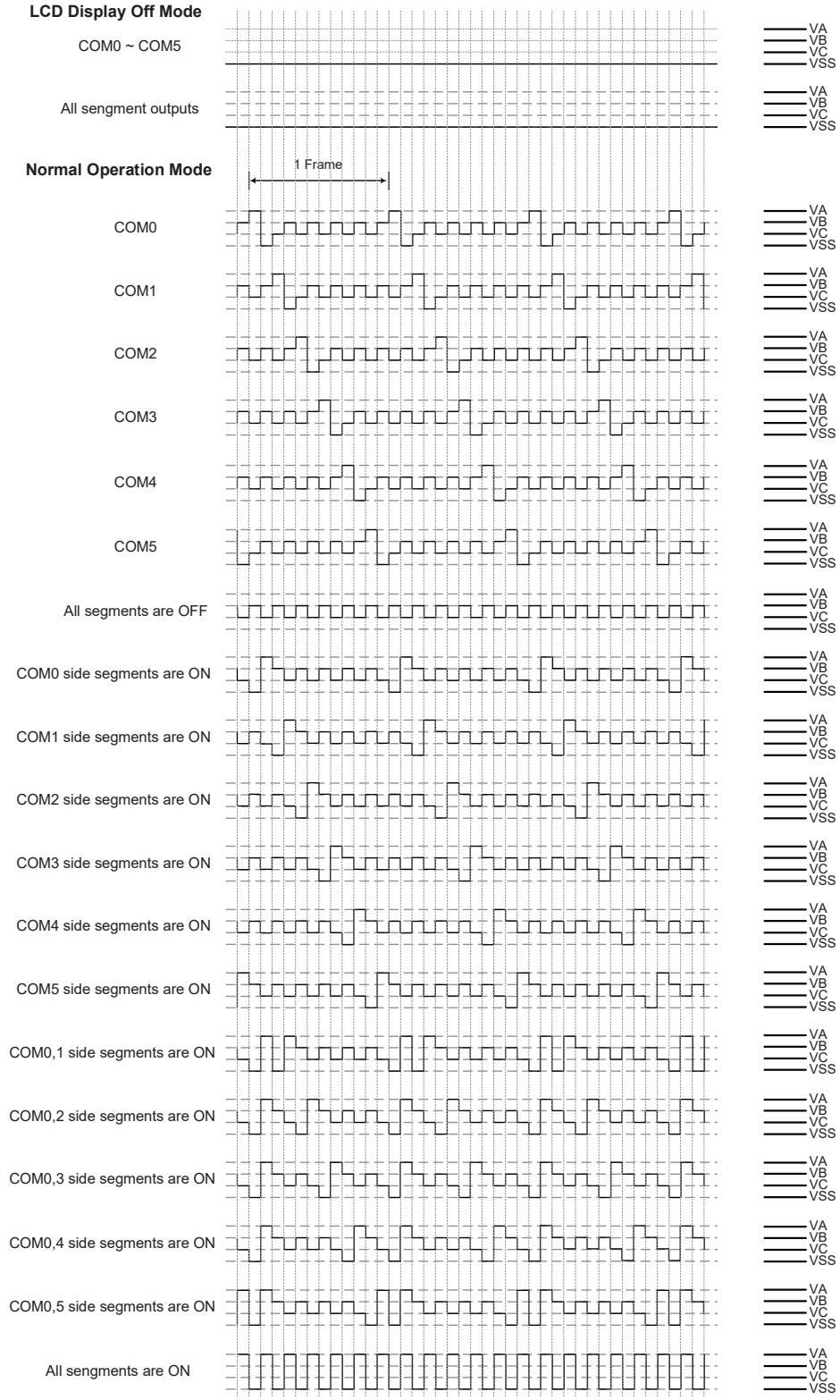


LCD Driver Output – Type A, 1/4 Duty, 1/3 Bias

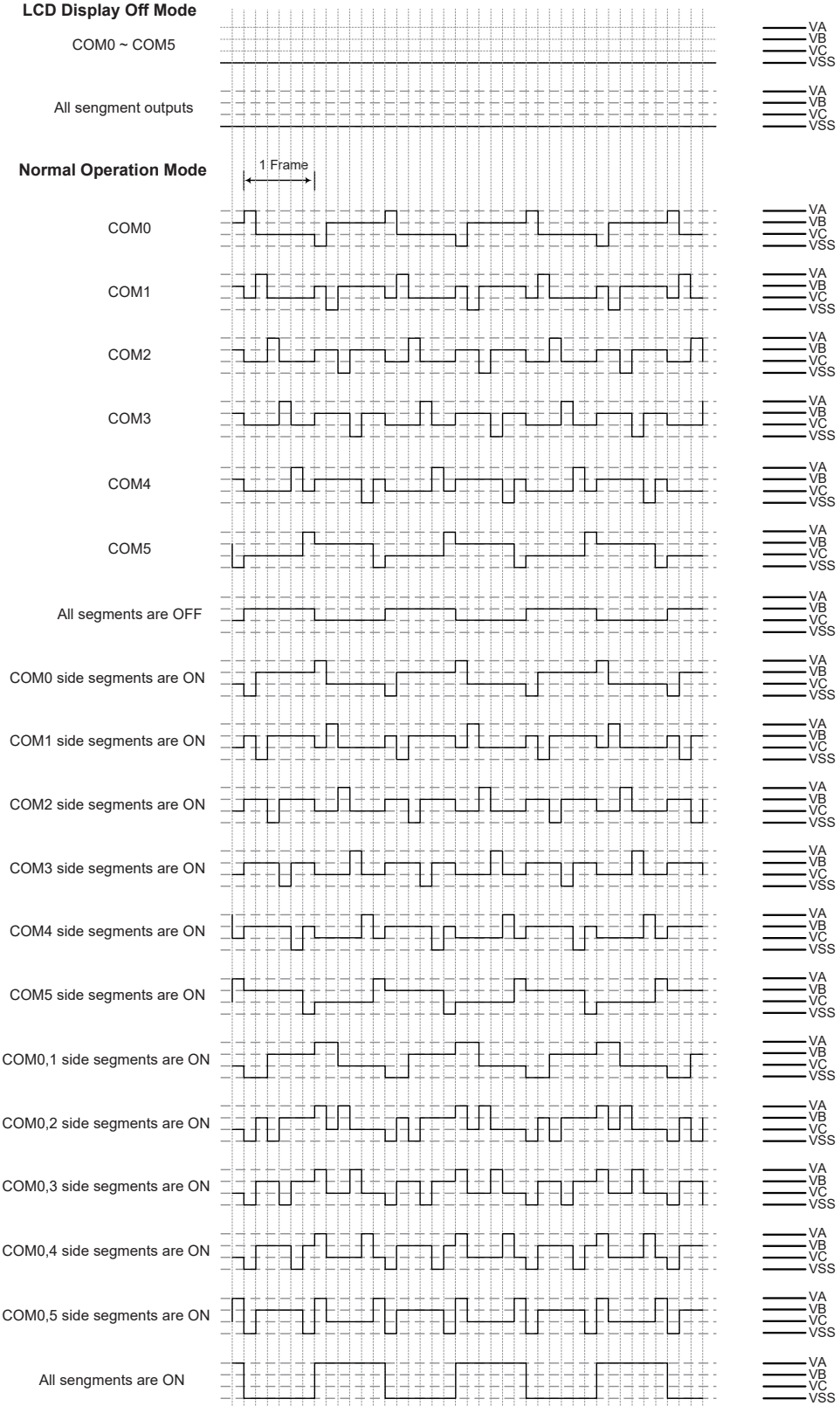


LCD Driver Output – Type B, 1/4 Duty, 1/3 Bias

R & C Type, 6-COM, 1/3 Bias

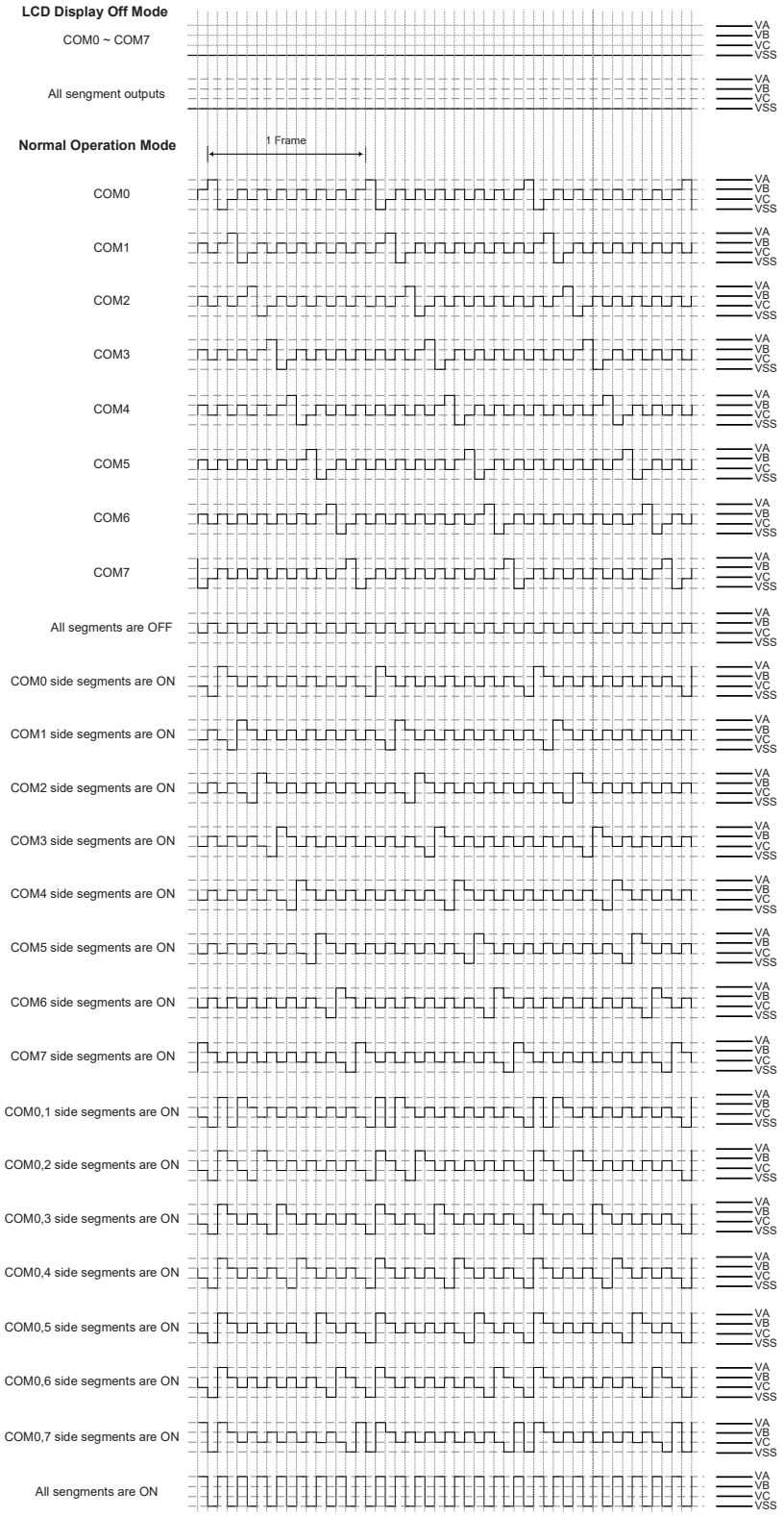


LCD Driver Output – Type A, 1/6 Duty, 1/3 Bias

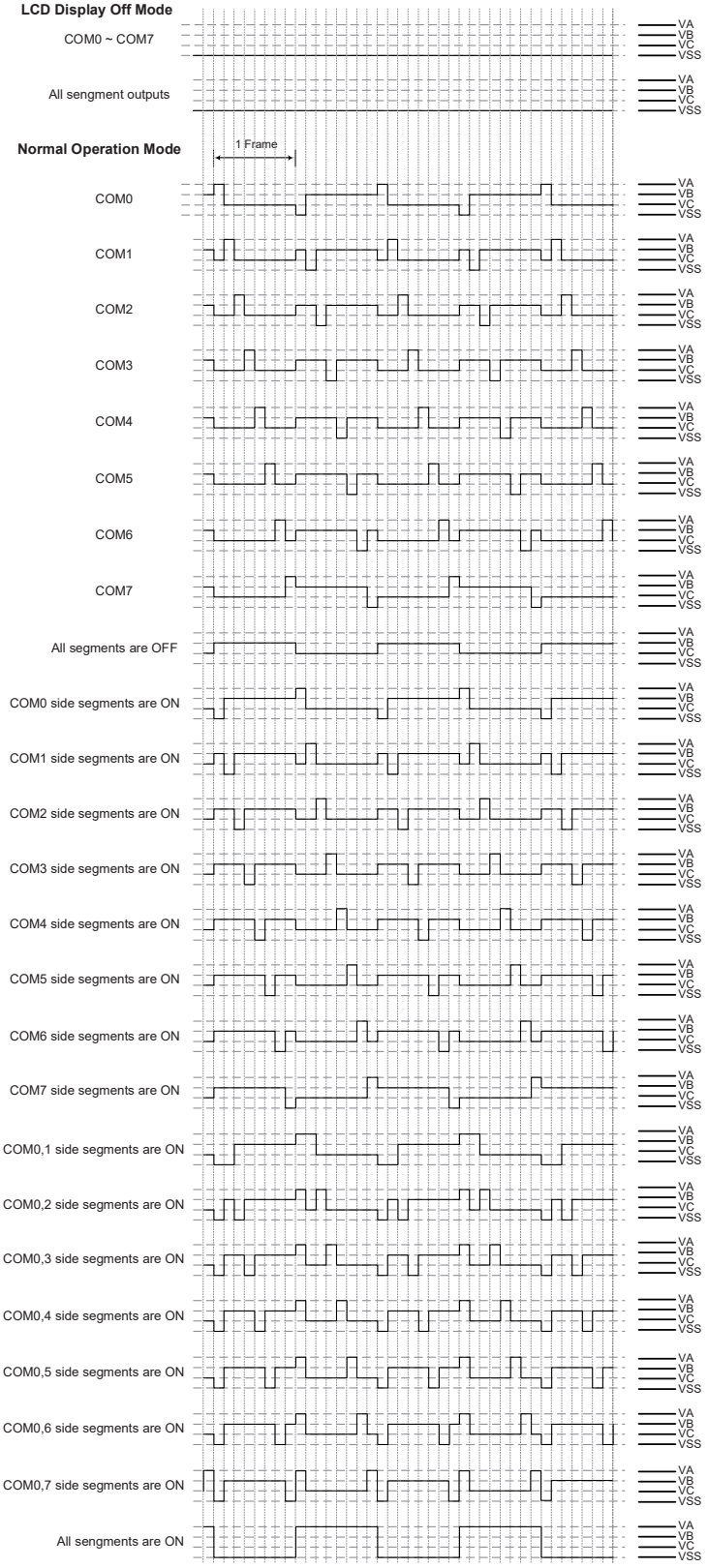


LCD Driver Output – Type B, 1/6 Duty, 1/3 Bias

R Type, 8-COM, 1/3 Bias

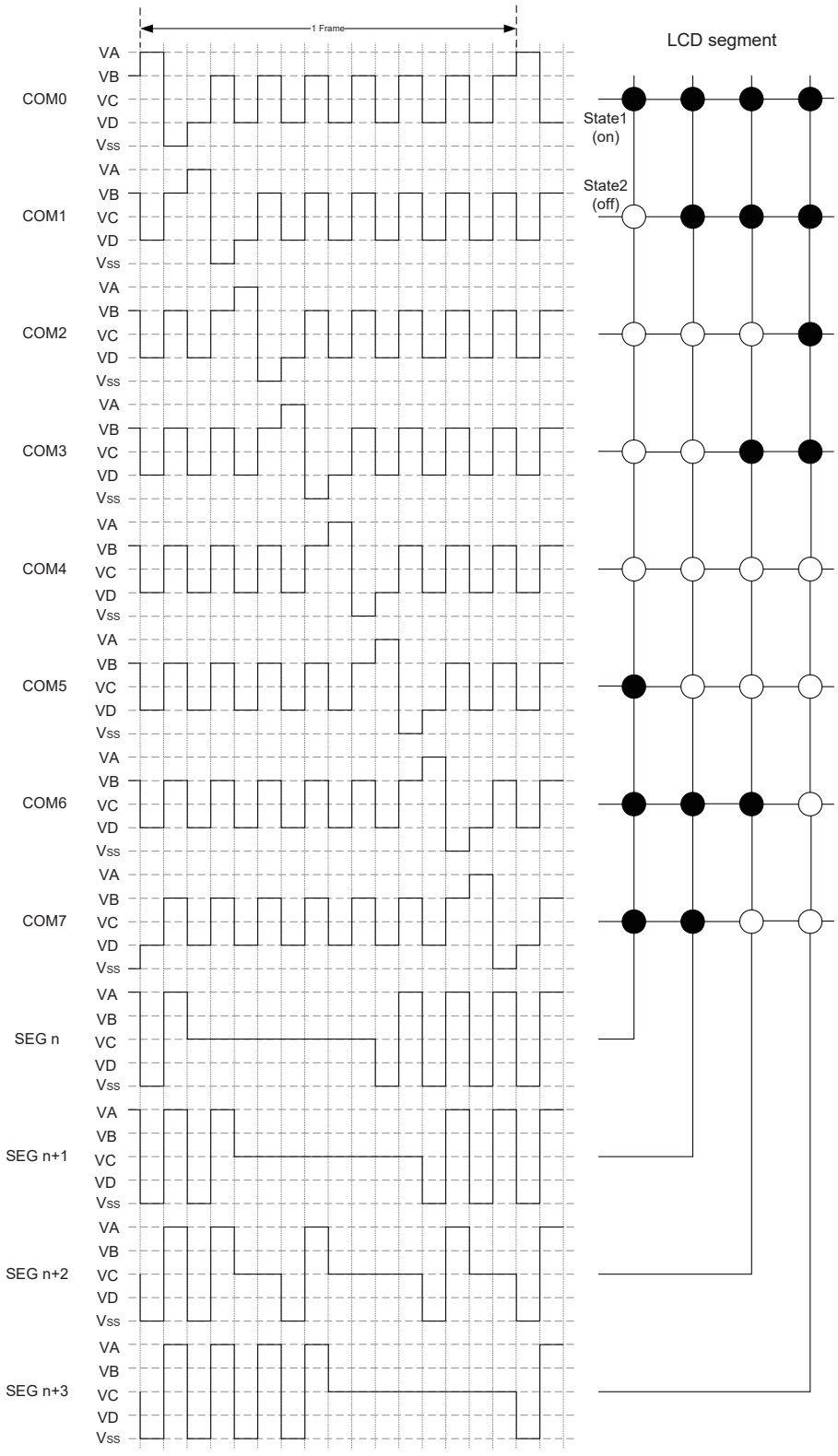


LCD Driver Output – Type A, 1/8 Duty, 1/3 Bias

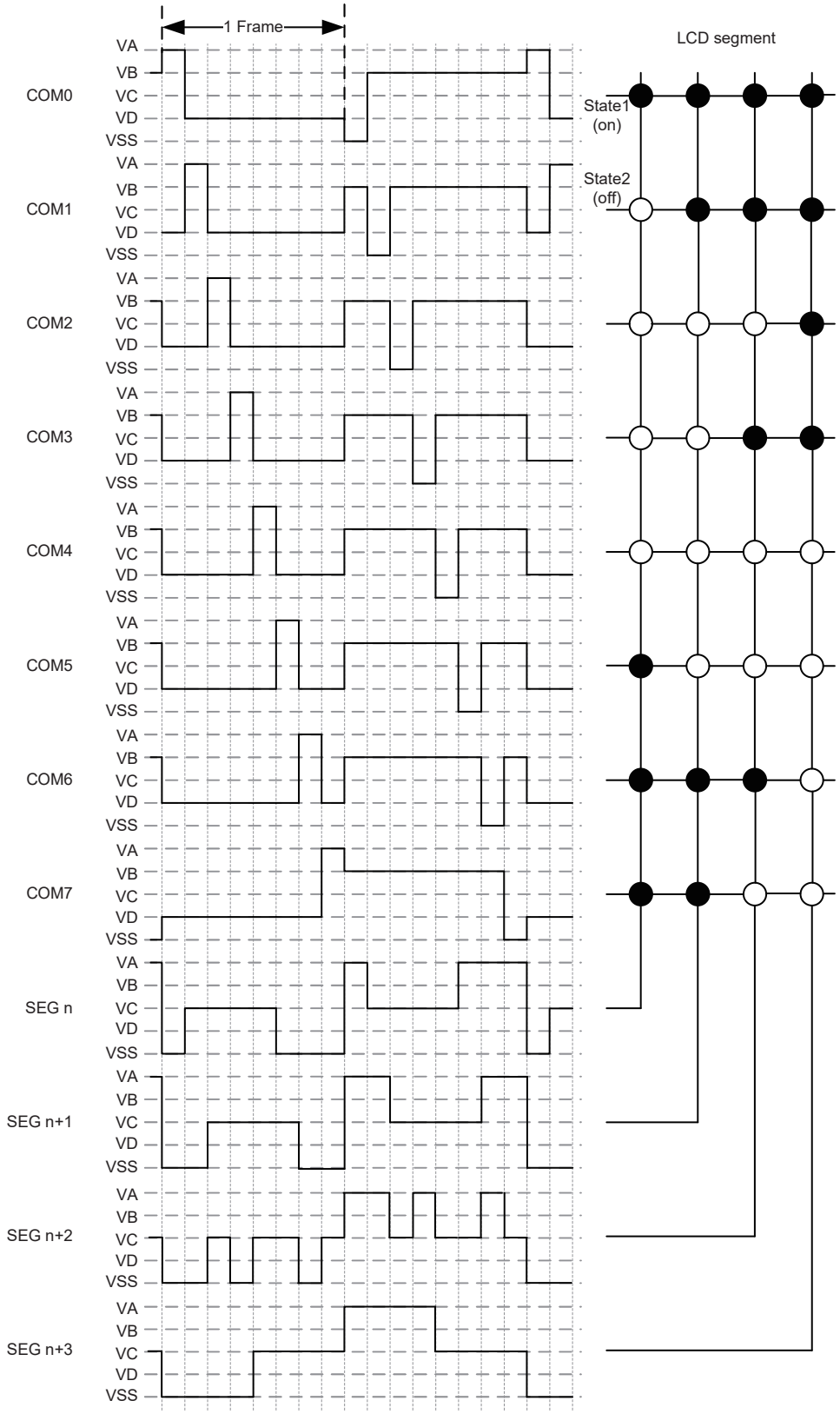


LCD Driver Output – Type B, 1/8 Duty, 1/3 Bias

R Type, 8-COM, 1/4 Bias



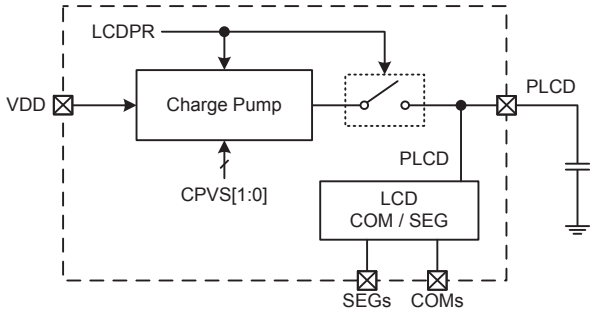
LCD Driver Output – Type A, 1/8 Duty, 1/4 Bias



LCD Driver Output – Type B, 1/8 Duty, 1/4 Bias

LCD Charge Pump

For the R type LCD, the COMs and SEGs pins can be powered up by the external PLCD pin or internal charge pump circuit which is determined by the LCDPR bit in the LCDCP register. When the LCDPR bit is set low, the LCD driver power is supplied by the external PLCD pin. If the LCDPR bit is set high, the LCD driver power is supplied by the internal charge pump circuit. There are four charge pump output voltage levels which are selected by the CPVS1~CPVS0 bits in the LCDCP register. If the internal charge pump circuit is used, an external 4.7μF capacitor should be connected to the external PLCD pin for output voltage stability. In addition, when using the C type LCD, the LCDPR bit should be fixed at 0.



R Type LCD Driver Charge Pump Circuit

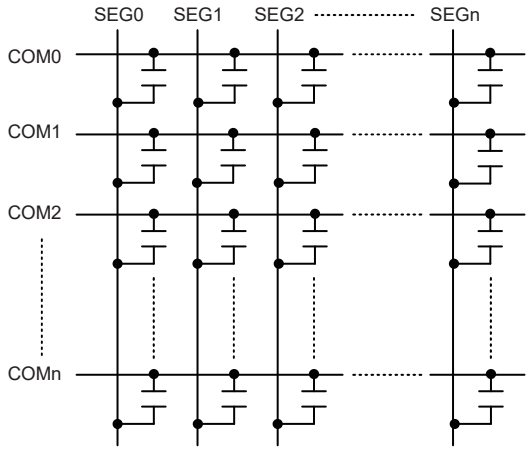
Programming Considerations

Certain precautions must be taken when programming the LCD. One of these is to ensure that the LCD Memory is properly initialised after the microcontroller is powered on. Like the General Purpose Data Memory, the contents of the LCD Memory are in an unknown condition after power-on. As the contents of the LCD Memory will be mapped into the actual display, it is important to initialise this memory area into a known condition soon after applying power to obtain a proper display pattern.

Consideration must also be given to the capacitive load of the actual LCD used in the application. As the load presented to the microcontroller by LCD pixels can be generally modeled as mainly capacitive in nature, it is important that this is not excessive, a point that is particularly true in the case of the COM lines which may be connected to many LCD pixels. The accompanying diagram depicts the equivalent circuit of the LCD.

One additional consideration that must be taken into account is what happens when the microcontroller enters the IDLE or SLOW Mode. The LCDEN control bit in the LCDC0 register permits the display to be powered off to reduce power consumption. If this bit is zero, the driving signals to the display will cease, producing a blank display pattern but reducing any power consumption associated with the LCD.

After Power-on, note that as the LCDEN bit is cleared to zero, the display function will be disabled.



LCD Panel Equivalent Circuit

Universal Serial Interface Module – USIM

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

SPI Interface

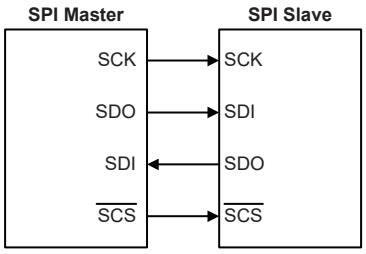
The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the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

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

SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I²C/UART function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. Communication between devices connected to the SPI interface

is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ pin only one slave device can be utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to 1 to enable $\overline{SCS}$ pin function, set CSEN bit to 0 the $\overline{SCS}$ pin will be floating state.

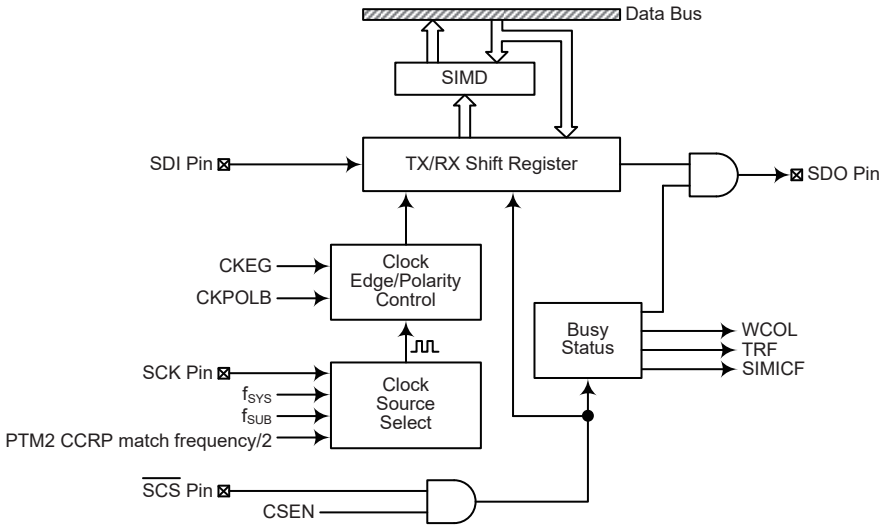


SPI Master/Slave Connection

The SPI function in the device offers the following features:

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

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



SPI Block Diagram

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two control registers, SIMC0 and SIMC2. Note that the SIMC2 and SIMD registers and their POR values are only available when the SPI mode is selected by properly configuring the UMD and SIM2~SIM0 bits in the SIMC0 register.

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

SPI Register List

SPI Data Register

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

- **SIMD Register**

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

"x": unknown

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

SPI Control Registers

There are also two control registers for the SPI interface, SIMC0 and SIMC2. The SIMC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC2 register is used for other control functions such as LSB/MSB selection, write collision flag etc.

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C 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 PTM2 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode

When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from PTM2 and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UMD**: UART mode selection bit
0: SPI or I²C mode
1: UART mode
This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I²C mode.

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
These bits are only available when the USIM is configured to operate in the I²C mode. Refer to the I²C register section.

Bit 1 **SIMEN**: USIM SPI/I²C Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the USIM SPI/I²C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I²C interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I²C interface is enabled. If the USIM is configured to operate as an SPI interface via the UMD and SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the USIM is configured to operate as an I²C interface via the UMD and SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

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

• SIMC2 Register

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

Bit 7~6 **D7~D6**: Undefined bits
These bits can be read or written by the application program.

Bit 5 **CKPOLB**: SPI clock line base condition selection
0: The SCK line will be high when the clock is inactive
1: The SCK line will be low when the clock is inactive
The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

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

CKPOLB=0
0: SCK is high base level and data capture at SCK rising edge
1: SCK is high base level and data capture at SCK falling edge

CKPOLB=1
0: SCK is low base level and data capture at SCK falling edge
1: SCK is low base level and data capture at SCK rising edge

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

Bit 3 **MLS**: SPI data shift order
0: LSB first
1: MSB first

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

Bit 2 **CSEN**: SPI $\overline{\text{SCS}}$ pin control
0: Disable
1: Enable

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

Bit 1 **WCOL**: SPI write collision flag
0: No collision
1: Collision

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

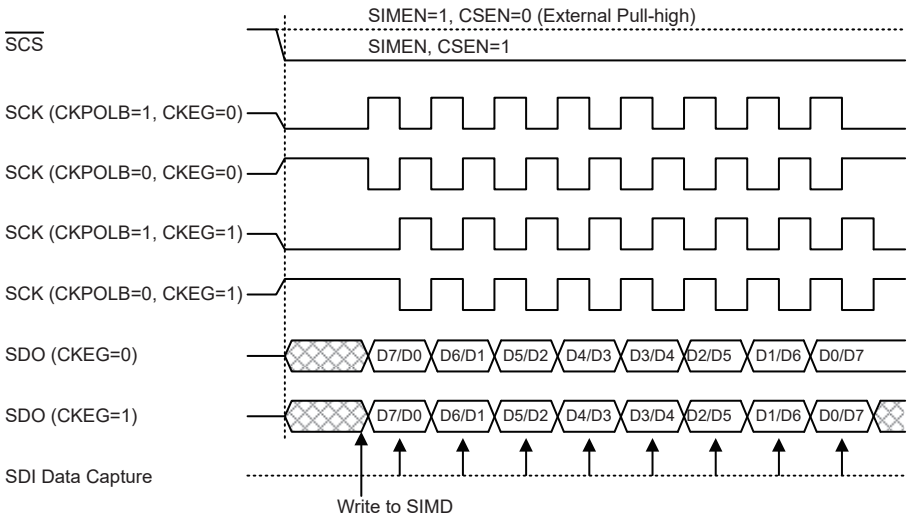
Bit 0 **TRF**: SPI Transmit/Receive complete flag
0: SPI data is being transferred
1: SPI data transmission is completed

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

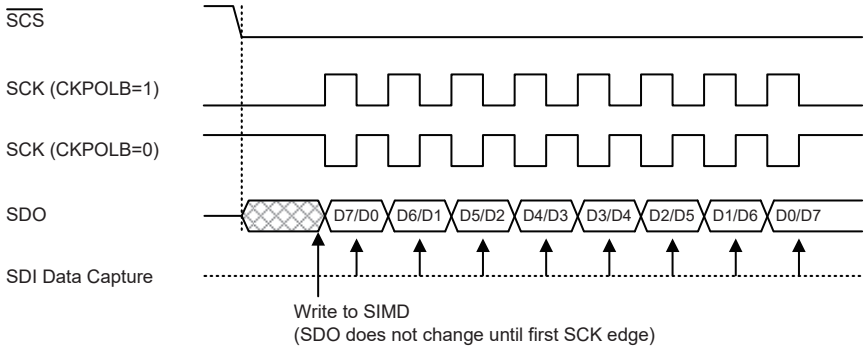
SPI Communication

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

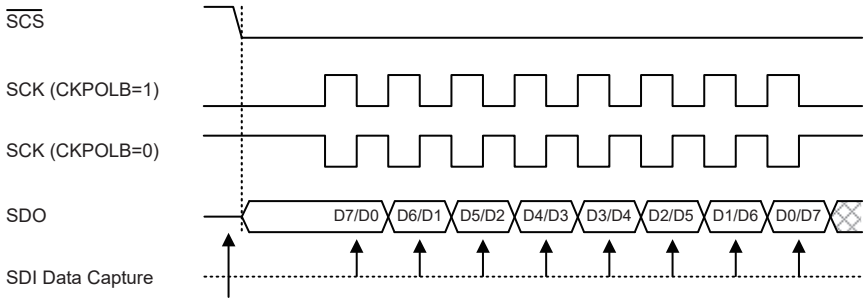
The SPI will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.



SPI Master Mode Timing

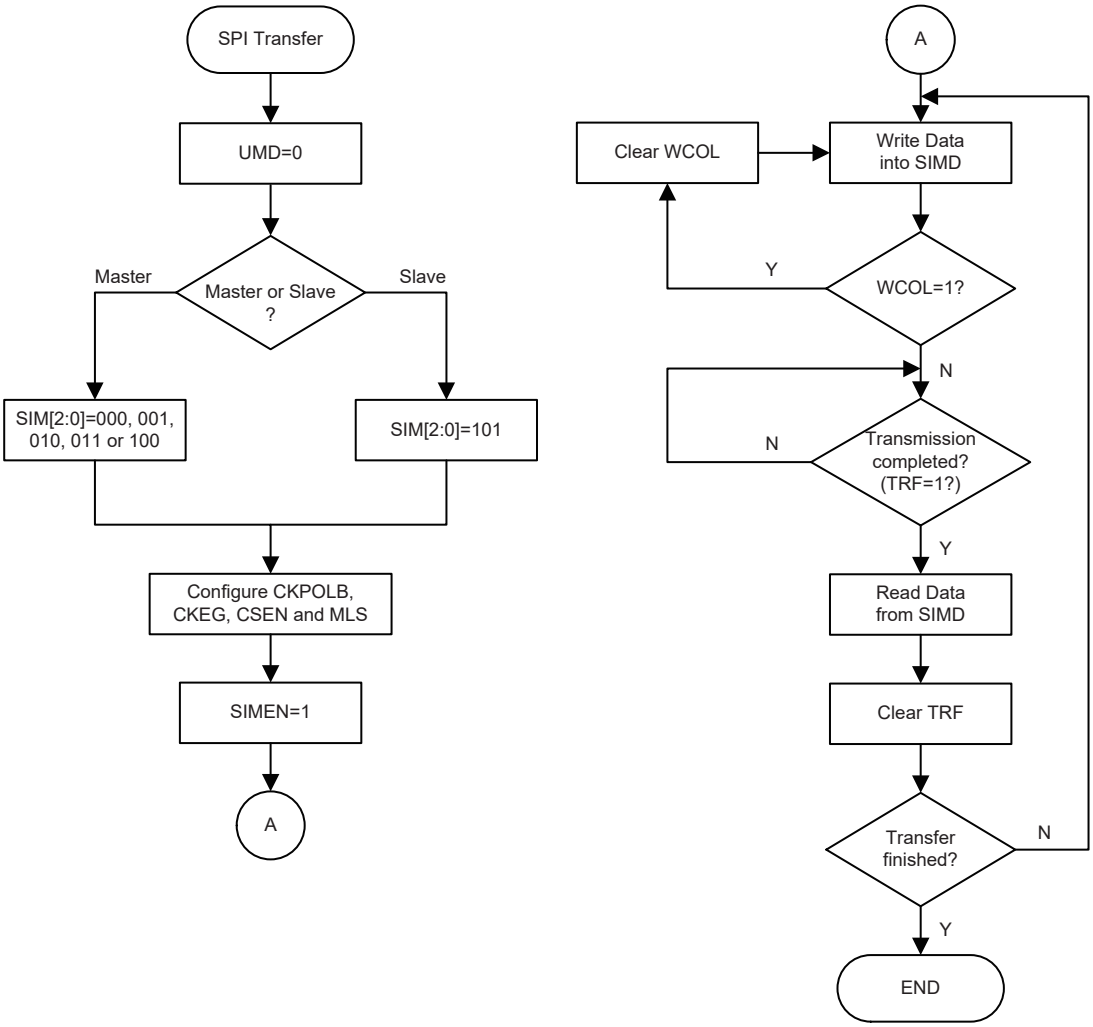


SPI Slave Mode Timing – CKEG=0



Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignores the $\overline{SCS}$ level.

SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flowchart

SPI Bus Enable/Disable

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

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

SPI Operation Steps

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

The CSEN bit in the SIMC2 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{SCS}$ line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the $\overline{SCS}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the SDI line in a

floating condition and the SDO line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and $\overline{SCS}$, SDI, SDO and SCK will all become I/O pins or the other functions using the corresponding pin-shared control bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SIMD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode:

- Step 1
  Select the SPI Master mode and clock source using the UMD and SIM2~SIM0 bits in the SIMC0 control register.
- Step 2
  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave devices.
- Step 3
  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then use the SCK and $\overline{SCS}$ lines to output the data. After this, go to step 5.
  For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.
- Step 5
  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the TRF bit or wait for a USIM SPI serial bus interrupt.
- Step 7
  Read data from the SIMD register.
- Step 8
  Clear TRF.
- Step 9
  Go to step 4.

Slave Mode:

- Step 1
  Select the SPI Slave mode using the UMD and SIM2~SIM0 bits in the SIMC0 control register
- Step 2
  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.
- Step 3
  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and $\overline{SCS}$ signal. After this, go to step 5.

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

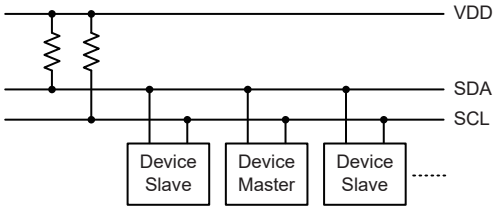
- Step 5
  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the TRF bit or wait for a USIM SPI serial bus interrupt.
- Step 7
  Read data from the SIMD register.
- Step 8
  Clear TRF.
- Step 9
  Go to step 4.

Error Detection

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

I²C Interface

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

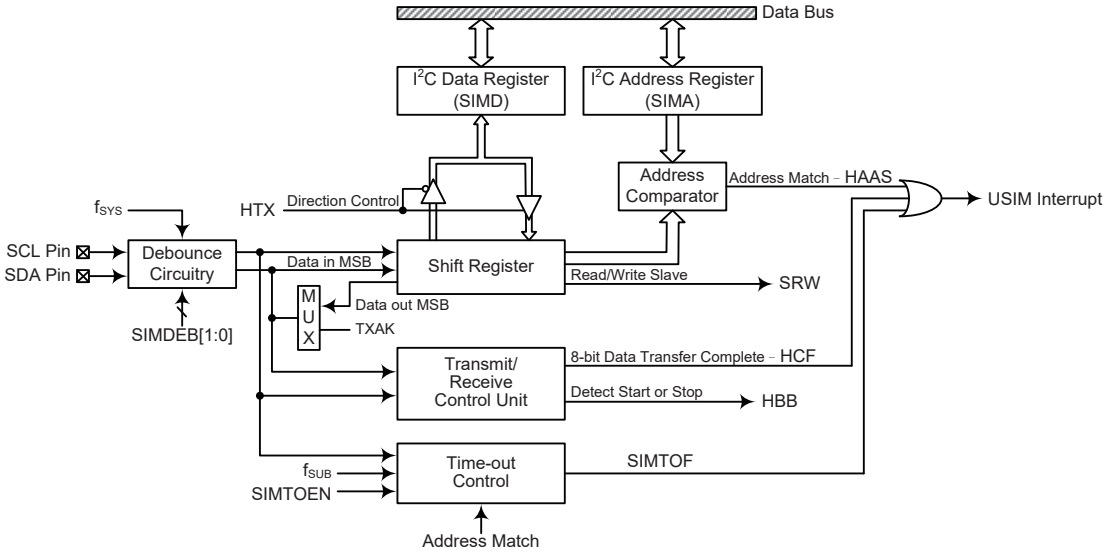


I²C Master Slave Bus Connection

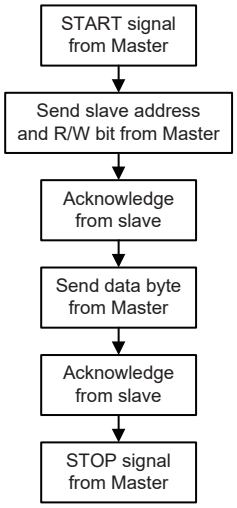
I²C Interface Operation

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

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



I²C Block Diagram



I²C Interface Operation

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

I²C Debounce Time Selection	I²C Standard Mode (100kHz)	I²C Fast Mode (400kHz)
No Debounce	f_{SYS} > 2MHz	f_{SYS} > 5MHz
2 system clock debounce	f_{SYS} > 4MHz	f_{SYS} > 10MHz
4 system clock debounce	f_{SYS} > 8MHz	f_{SYS} > 20MHz

I²C Minimum f_{SYS} Frequency Requirements

I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMTOC, one address register SIMA and one data register, SIMD. Note that the SIMC1, SIMD, SIMA and SIMTOC registers and their POR values are only available when the I²C mode is selected by properly configuring the UMD and SIM2~SIM0 bits in the SIMC0 register.

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

I²C Register List

I²C Data Register

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

- **SIMD Register**

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

"x": unknown

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

I²C Address Register

The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the SIMA register define the device slave address. Bit 0 is not defined. When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected.

- **SIMA Register**

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

Bit 7~1 **SIMA6~SIMA0**: I²C slave address
SIMA6~SIMA0 is the I²C slave address bit 6 ~ bit 0.

Bit 0 **D0**: Reserved bit, can be read or written

I²C Control Registers

There are three control registers for the I²C interface, SIMC0, SIMC1 and SIMTOC. The SIMC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, SIMTOC, is used to control the I²C time-out function and is described in the corresponding section.

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C 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 PTM2 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode

When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from PTM2 and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UMD**: UART mode selection bit
0: SPI or I²C mode
1: UART mode

This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I²C mode.

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
1x: 4 system clock debounce

These bits are used to select the I²C debounce time when the USIM is configured as the I²C interface function by setting the UMD bit to "0" and the SIM2~SIM0 bits to "110".

Bit 1 **SIMEN**: USIM SPI/I²C Enable Control
0: Disable
1: Enable

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

at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: USIM SPI Incomplete Flag
This bit is only available when the USIM is configured to operate in an SPI slave mode. Refer to the SPI register section.

• **SIMC1 Register**

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

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

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

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

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

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

Bit 2 **SRW**: I²C Slave Read/Write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode
The SRW flag is the I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C bus. When the transmitted address and slave address is match, that is when the HAAS flag is set high, the slave device will check the SRW flag to determine whether it should be in transmit mode or receive mode. If the SRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

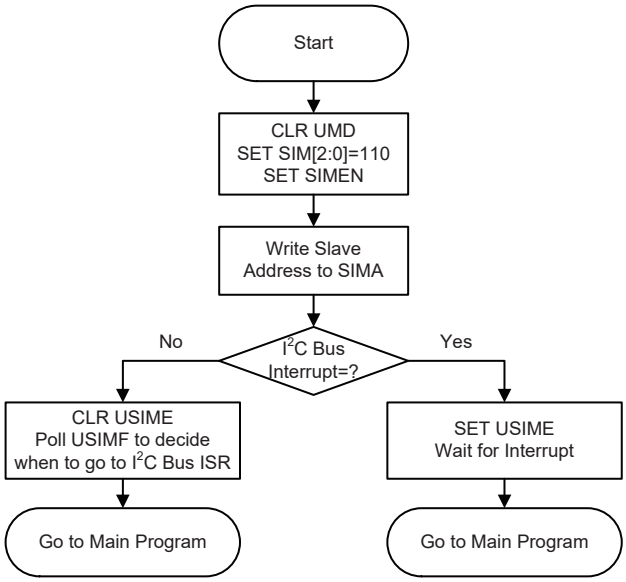
Bit 1 **IAMWU**: I²C Address Match Wake-up control
0: Disable
1: Enable
This bit should be set to 1 to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake up, then this bit must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

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

I²C Bus Communication

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the SIMC1 register will be set and an USIM interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer completion or from the I²C bus time-out occurrence. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
  Set the UMD, SIM2~SIM0 and SIMEN bits in the SIMC0 register to "0", "110" and "1" respectively to enable the I²C bus.
- Step 2
  Write the slave address of the device to the I²C bus address register SIMA.
- Step 3
  Set the USIME interrupt enable bit of the interrupt control register to enable the USIM interrupt.



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

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

I²C Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal USIM I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the SIMC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

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

I²C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the master device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is “1” then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the SRW flag is “0” then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

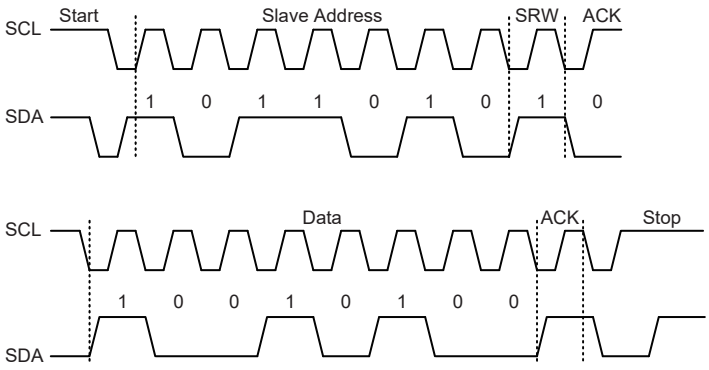
I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be set to "0".

I²C Bus Data and Acknowledge Signal

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

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the RXAK bit in the SIMC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.

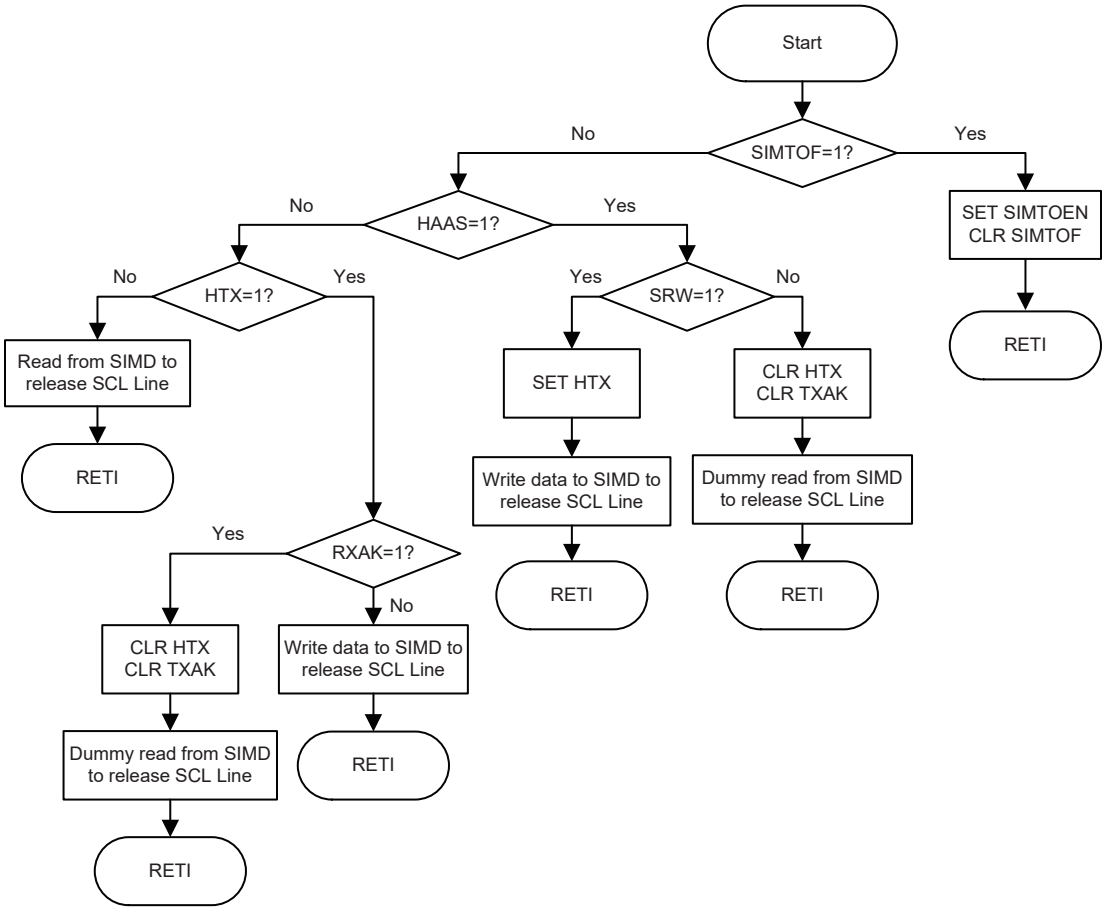


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

I²C Communication Timing Diagram

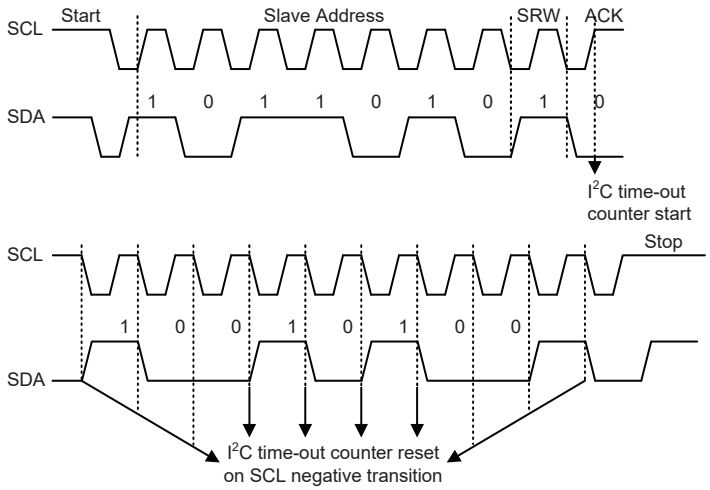
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.



I²C Bus ISR Flow Chart

I²C Time-out Control

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



I²C Time-out

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

Registers	After I²C Time-out
SIMD, SIMA, SIMC0	No change
SIMC1	Reset to POR condition

I²C Registers after Time-out

The SIMTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using SIMTOS bit field in the SIMTOC register. The time-out time is given by the formula: ((1~64)×32)/f_{SUB}. This gives a time-out period which ranges from about 1ms to 64ms.

- **SIMTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

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

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

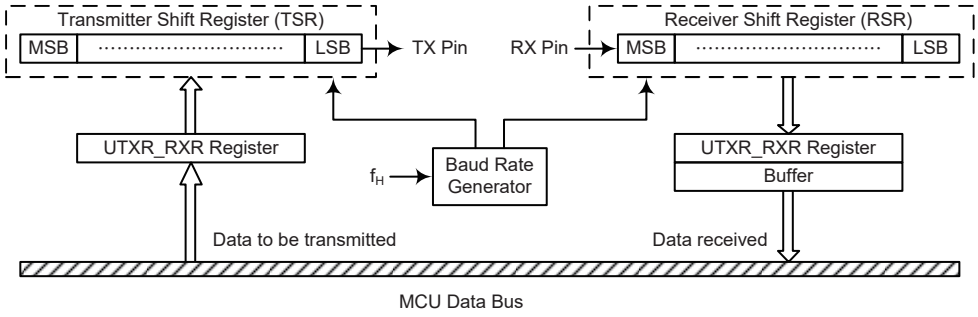
Bit 5~0 **SIMTOS5~SIMTOS0**: USIM I²C Time-out period selection
I²C time-out clock source is f_{SUB}/32.
I²C time-out time is equal to (SIMTOS[5:0]+1)×(32/f_{SUB}).

UART Interface

The device contains 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 shares the same internal interrupt vector with the SPI and I^2C interfaces 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 initialized 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 UMD bit, the UREN bit, the UTXEN and URXEN 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 UMD, UREN, UTXEN or URXEN 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 UTXR_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 UTXR_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 UTXR_RXR register, where it is buffered and can be manipulated by the application program. Only the UTXR_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 UTXR_RXR register is used for both data transmission and data reception.

UART Status and Control Registers

There are six control registers associated with the UART function. The UMD bit in the SIMC0 register can be used to select the UART mode. The UUSR, UUCR1 and UUCR2 registers control the overall function of the UART, while the UBRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the UTXR_RXR data register. Note that UART related registers and their POR values are only available when the UART mode is selected by setting the UMD bit in the SIMC0 register to "1".

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
UUSR	UPERR	UNF	UFERR	UOERR	URIDLE	URXIF	UTIDLE	UTXIF
UUCR1	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
UUCR2	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
UTXR_RXR	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
UBRG	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0

UART Register List

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I^2C Operating Mode Control

When the UMD bit is cleared to zero, these bits setup the SPI or I^2C operating mode of the USIM function. Refer to the SPI or I^2C register section for more details.

Bit 4 **UMD**: UART mode selection bit
0: SPI or I²C mode
1: UART mode
This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I²C mode.

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
Refer to the I²C register section.

Bit 1 **SIMEN**: USIM SPI/I²C Enable Control
This bit is only available when the USIM is configured to operate in an SPI or I²C mode with the UMD bit set low. Refer to the SPI or I²C register section for more details.

Bit 0 **SIMICF**: USIM SPI Incomplete Flag
Refer to the SPI register section.

• UUSR Register

The UUSR 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 UUSR register are read only. Further explanation on each of the flags is given below:

Bit	7	6	5	4	3	2	1	0
Name	UPERR	UNF	UFERR	UOERR	URIDLE	URXIF	UTIDLE	UTXIF
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

Bit 7 **UPERR**: Parity error flag
0: No parity error is detected
1: Parity error is detected
The UPERR 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 by a software sequence which involves a read to the status register UUSR followed by an access to the UTXR_RXR data register.

Bit 6 **UNF**: Noise flag
0: No noise is detected
1: Noise is detected
The UNF 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 UNF flag is set during the same cycle as the URXIF flag but will not be set in the case of as overrun. The UNF flag can be cleared by a software sequence which will involve a read to the status register UUSR followed by an access to the UTXR_RXR data register.

Bit 5 **UFERR**: Framing error flag
0: No framing error is detected
1: Framing error is detected
The UFERR 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 by a software sequence which will involve a read to the status register UUSR followed by an access to the UTXR_RXR data register.

Bit 4 **UOERR**: Overrun error flag
0: No overrun error is detected
1: Overrun error is detected
The UOERR 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 UTXR_RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register UUSR followed by an access to the UTXR_RXR data register.

Bit 3 **URIDLE**: Receiver status
0: Data reception is in progress (Data being received)
1: No data reception is in progress (Receiver is idle)
The URIDLE 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 URIDLE bit is "1" indicating that the UART receiver is idle and the RX pin stays in logic high condition.

Bit 2 **URXIF**: Receive UTXR_RXR data register status
0: UTXR_RXR data register is empty
1: UTXR_RXR data register has available data
The URXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the UTXR_RXR read data register is empty. When the flag is "1", it indicates that the UTXR_RXR read data register contains new data. When the contents of the shift register are transferred to the UTXR_RXR register, an interrupt is generated if URIE=1 in the UUCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags UNF, UFERR, and/or UPERR are set within the same clock cycle. The URXIF flag will eventually be cleared when the UUSR register is read with URXIF set, followed by a read from the UTXR_RXR register, and if the UTXR_RXR register has no more new data available.

Bit 1 **UTIDLE**: Transmission idle
0: Data transmission is in progress (Data being transmitted)
1: No data transmission is in progress (Transmitter is idle)
The UTIDLE 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 UTXIF flag is "1" and when there is no transmit data or break character being transmitted. When UTIDLE is equal to "1", the TX pin becomes idle with the pin state in logic high condition. The UTIDLE flag is cleared by reading the UUSR register with UTIDLE set and then writing to the UTXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 **UTXIF**: Transmit UTXR_RXR data register status
0: Character is not transferred to the transmit shift register
1: Character has transferred to the transmit shift register (UTXR_RXR data register is empty)
The UTXIF 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 UTXR_RXR data register. The UTXIF flag is cleared by reading the UART status register (UUSR) with UTXIF set and then writing to the UTXR_RXR data register. Note that when the UTXEN bit is set, the UTXIF flag bit will also be set since the transmit data register is not yet full.

• UUCR1 Register

The UUCR1 register together with the UUCR2 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	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
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 **UREN**: 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 UREN 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 if the UMD bit is set and the TX and RX pins will function as defined by the UTXEN and URXEN 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 UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF bits will be cleared, while the UTIDLE, UTXIF and URIDLE bits will be set. Other control bits in UUCR1, UUCR2 and UBRG registers will remain unaffected. If the UART is active and the UREN bit is cleared, 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 **UBNO**: 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 URX8 and UTX8 will be used to store the 9th bit of the received and transmitted data respectively.

Bit 5 **UPREN**: 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 **UPRT**: 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 **USTOPS**: 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 **UTXBRK**: Transmit break character
0: No break character is transmitted
1: Break characters transmit

The UTXBRK 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 UTXBRK bit is reset.

Bit 1 **URX8**: 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 URX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **UTX8**: 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 UTX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

- **UUCR2 Register**

The UUCR2 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 USIM UART mode 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	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
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 **UTXEN**: UART Transmitter enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled
The bit named UTXEN 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 UTXEN bit is equal to "1" and the UMD and UREN bit are also equal to "1", the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the UTXEN 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 **URXEN**: UART Receiver enabled control
0: UART receiver is disabled
1: UART receiver is enabled
The bit named URXEN 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 URXEN bit is equal to "1" and the UMD and UREN bit are also equal to "1", the receiver will be enabled and the RX pin will be controlled by the UART. Clearing the URXEN 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 **UBRGH**: Baud Rate speed selection
0: Low speed baud rate
1: High speed baud rate
The bit named UBRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register UBRG, 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 **UADDEN**: Address detect function enable control
0: Address detect function is disabled
1: Address detect function is enabled

The bit named UADDEN 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 URX7 if UBNO=0 or the 9th bit, which corresponds to URX8 if UBNO=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 UBNO. 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 **UWAKE**: 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 UWAKE bit is set to 1 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 UWAKE bit is cleared to 0.

Bit 2 **URIE**: 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 UOERR or receive data available flag URXIF is set, the USIM interrupt request flag USIMF will be set. If this bit is equal to "0", the USIM interrupt request flag USIMF will not be influenced by the condition of the UOERR or URXIF flags.

Bit 1 **UTIIE**: 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 UTIDLE is set, due to a transmitter idle condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to "0", the USIM interrupt request flag USIMF will not be influenced by the condition of the UTIDLE flag.

Bit 0 **UTEIE**: 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 UTXIF is set, due to a transmitter empty condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to "0", the USIM interrupt request flag USIMF will not be influenced by the condition of the UTXIF flag.

• **UTXR_RXR Register**

The UTXR_RXR register is the data register which is used to store the data to be transmitted on the TX pin or being received from the RX pin.

Bit	7	6	5	4	3	2	1	0
Name	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
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 **UTXRX7~UTXRX0**: UART Transmit/Receive Data bit 7 ~ bit 0

• **UBRG Register**

Bit	7	6	5	4	3	2	1	0
Name	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0
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 **UBRG7~UBRG0**: Baud Rate values

By programming the UBRGH bit in UUCR2 Register which allows selection of the related formula described above and programming the required value in the UBRG register, the required baud rate can be setup.

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

Baud rate=f_H/[16×(N+1)] if UBRGH=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 UBRG and the second is the value of the UBRGH bit with the control register UUCR2. The UBRGH 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 UBRG 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 UBRG register and has a range of between 0 and 255.

UUCR2 UBRGH Bit	0	1
Baud Rate (BR)	f_H/[64(N+1)]	f_H/[16(N+1)]

By programming the UBRGH bit which allows selection of the related formula and programming the required value in the UBRG 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 UBRG register, there will be an error associated between the actual and requested value. The following example shows how the UBRG 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 UBRGH cleared to zero determine the UBRG 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×64)]–1

Giving a value for N=[4000000/(4800×64)]–1=12.0208

To obtain the closest value, a decimal value of 12 should be placed into the UBRG register. This gives an actual or calculated baud rate value of BR=4000000/[64×(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 UBNO, UPRT, UPREN, and USTOPS bits in the UUCR1 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 UREN bit in the UUCR1 register. When the UART mode is selected by setting the UMD bit in the SIMC0 register to "1", if the UREN, UTXEN and URXEN 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 UREN 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 UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF being cleared while bits UTIDLE, UTXIF and URIDLE will be set. The remaining control bits in the UUCR1, UUCR2 and UBRG registers will remain unaffected. If the UREN bit in the UUCR1 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.

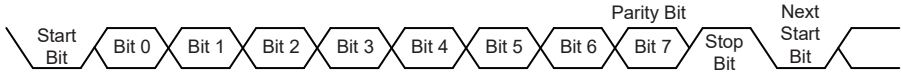
Data, Parity and Stop Bit Selection

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 UUCR1 register. The UBNO bit controls the number of data bits which can be set to either 8 or 9, the UPRT bit controls the choice of odd or even parity, the UPREN bit controls the parity on/off function and the USTOPS 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.

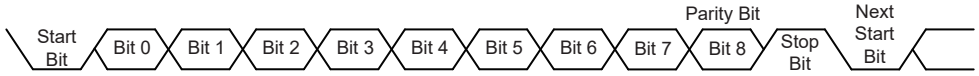
Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
Example of 9-bit Data Formats				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

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 UBNO bit in the UUCR1 register. When UBNO 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 UTX8 bit in the UUCR1 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 UTXR_RXR register. The data to be transmitted is loaded into this UTXR_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 UTXR_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 UTXEN bit is set, but the data will not be transmitted until the UTXR_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 UTXR_RXR register, after which the UTXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the UTXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the UTXEN 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 UTXR_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 UTX8 bit in the UUCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the UBNO, UPRT, UPREN and USTOPS bits to define the required word length, parity type and number of stop bits.
- Setup the UBRG register to select the desired baud rate.
- Set the UTXEN bit to ensure that the TX pin is used as a UART transmitter pin.
- Access the UUSR register and write the data that is to be transmitted into the UTXR_RXR register. Note that this step will clear the UTXIF bit.

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

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

1. A UUSR register access

2. A UTXR_RXR register write execution

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

During a data transmission, a write instruction to the UTXR_RXR register will place the data into the UTXR_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 UTXR_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the UTXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the UTIDLE bit will be set. To clear the UTIDLE bit the following software sequence is used:

1. A UUSR register access

2. A UTXR_RXR register write execution

Note that both the UTXIF and UTIDLE bits are cleared by the same software sequence.

Transmit Break

If the UTXBRK 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 UTXBRK 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 UTXBRK 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 UTXBRK 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 UBNO bit is set, the word length will be set to 9 bits with the MSB being stored in the URX8 bit of the UUCR1 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 UTXR_RXR register forms a buffer between the internal bus and the receiver shift register. The UTXR_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 UTXR_RXR before the third byte has been completely shifted in, otherwise this third byte will be discarded and an overrun error UOERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of UBNO, UPRT and UPREN bits to define the word length, parity type.
- Setup the UBRG register to select the desired baud rate.
- Set the URXEN 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 URXIF bit in the UUSR register will be set when the UTXR_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 UTXR_RXR register, then if the URIE 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 URXIF bit can be cleared using the following software sequence:

1. A UUSR register access
2. A UTXR_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 UBNO 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 UBNO plus one stop bit. The URXIF bit is set, UFERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the URIDLE bit is set. A break is regarded as a character that contains only zeros with the UFERR 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 UFERR 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 URIDLE 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, UFERR, will be set.
- The receive data register, UTXR_RXR, will be cleared.
- The UOERR, UNF, UPERR, URIDLE or URXIF 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 UUSR register, otherwise known as the URIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the URIDLE flag will have a high value, which indicates the receiver is in an idle condition.

Receiver Interrupt

The read only receive interrupt flag URXIF in the UUSR register is set by an edge generated by the receiver. An interrupt is generated if URIE=1, when a word is transferred from the Receive Shift Register, RSR, to the Receive Data Register, UTXR_RXR. An overrun error can also generate an interrupt if URIE=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 – UOERR

The UTXR_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 UTXR_RXR register. If this is not done, the overrun error flag UOERR will be consequently indicated.

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

- The UOERR flag in the UUSR register will be set.
- The UTXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the URIE bit is set.

The UOERR flag can be cleared by an access to the UUSR register followed by a read to the UTXR_RXR register.

Noise Error – UNF

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, UNF, in the UUSR register will be set on the rising edge of the URXIF bit.
- Data will be transferred from the Shift register to the UTXR_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the URXIF bit which itself generates an interrupt.

Note that the UNF flag is reset by a UUSR register read operation followed by a UTXR_RXR register read operation.

Framing Error – UFERR

The read only framing error flag, UFERR, in the UUSR 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 UFERR flag will be set. The UFERR flag and the received data will be recorded in the UUSR and UTXR_RXR registers respectively, and the flag is cleared in any reset.

Parity Error – UPERR

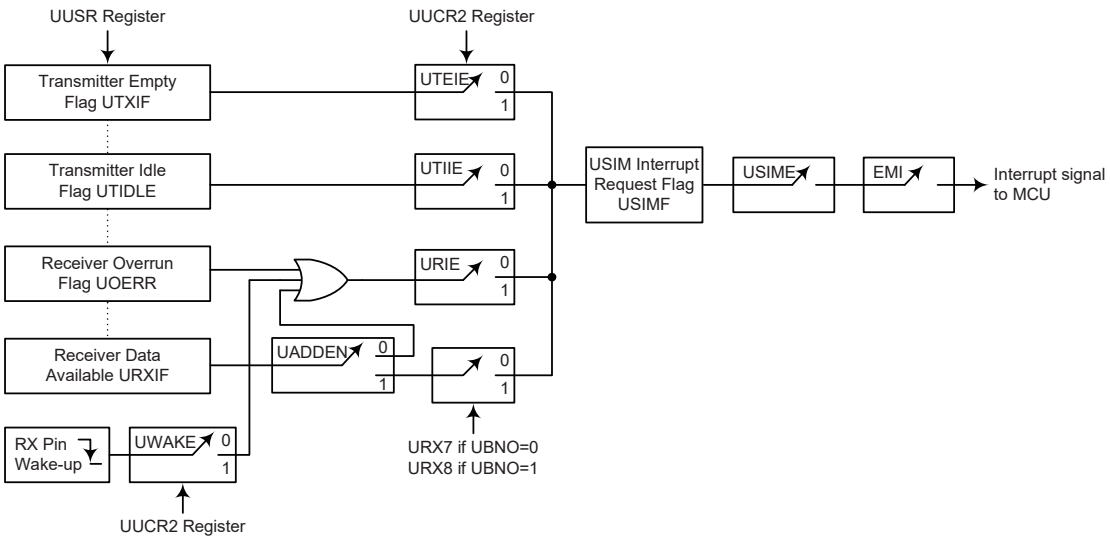
The read only parity error flag, UPERR, in the UUSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, UPREN=1, and if the parity type, odd or even is selected. The read only UPERR flag and the received data will be recorded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

UART Interrupt Structure

Several individual UART conditions can trigger an USIM 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 and the USIM interrupt control bit are 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 UUSR register flags which will generate an USIM interrupt if its associated interrupt enable control bit in the UUCR2 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 USIM UART mode interrupt sources.

The address detect condition, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt when an address detect condition occurs if its function is enabled by setting the UADDEN bit in the UUCR2 register. An RX pin wake-up, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt if the UART clock (f_H) source is switched off and the UWAKE and URIE bits in the UUCR2 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 UUSR 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 USIM interrupt enable control bit in the interrupt control register 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, UADDEN, in the UUCR2 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 URXIF flag. If the UADDEN 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 USIME and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if UBNO=1 or the 8th bit if UBNO=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 UADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the URXIF 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 UPREN to zero.

UADDEN	Bit 9 if UBNO=1 Bit 8 if UBNO=0	USIM Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

UADDEN 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 UUSR, UUCR1, UUCR2, transmit and receive registers, as well as the UBRG 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 UWAKE bit in the UUCR2 register. If this bit, along with the UART mode selection bit, UMD, the UART enable bit, UREN, the receiver enable bit, URXEN and the receiver interrupt bit, URIE, 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 USIM interrupt enable bit, USIME, must be set. If the EMI and USIME 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 USIM interrupt will not be generated until after this time has elapsed.

Serial Peripheral Interface – SPIA

These devices contain an independent SPI function. It is important not to confuse this independent SPI function with the additional one contained within the combined SIM function, which is described in another section of this datasheet. This independent SPI function will carry the name SPIA to distinguish it from the other one in the SIM.

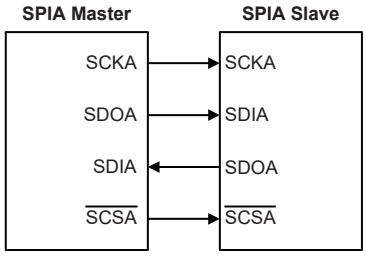
The SPIA interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four line SPIA interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the 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 SPIA interface specification can control multiple slave devices from a single master, however the device is provided with only one $\overline{\text{SCSA}}$ 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.

SPIA Interface Operation

The SPIA interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDIA, SDOA, SCKA and $\overline{\text{SCSA}}$. Pins SDIA and SDOA are the Serial Data Input and Serial Data Output lines, the SCKA pin is the Serial Clock line and $\overline{\text{SCSA}}$ is the Slave Select line. As the SPIA interface pins are pin-shared with normal I/O pins, the SPIA interface must first be enabled by configuring the corresponding selection bits in the pin-shared function selection registers. The SPIA can be disabled or enabled using the SPIAEN bit in the SPIAC0 register. Communication between devices connected to the SPIA interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{\text{SCSA}}$ pin only one slave device can be utilized.

The $\overline{\text{SCSA}}$ pin is controlled by the application program, set the SACSEN bit to "1" to enable the $\overline{\text{SCSA}}$ pin function and clear the SACSEN bit to "0" to place the $\overline{\text{SCSA}}$ pin into a floating state.

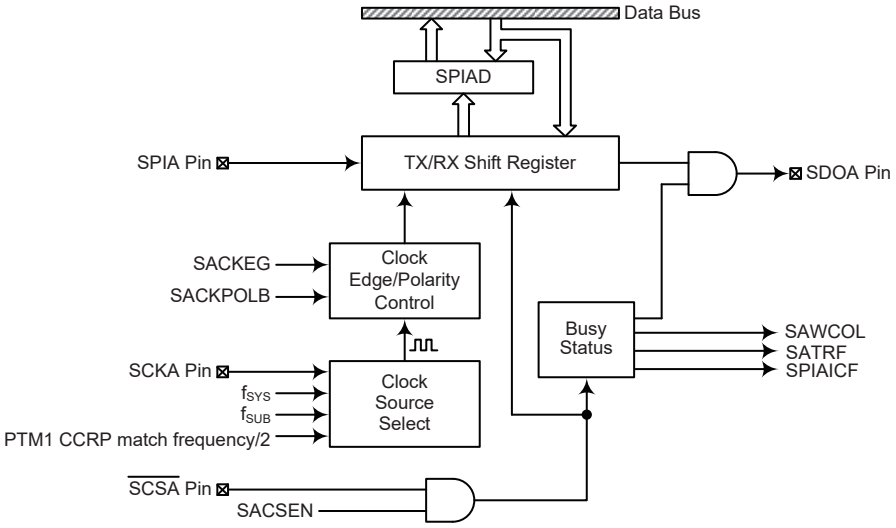


SPIA Master/Slave Connection

The SPIA function in the device offers the following features:

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

The status of the SPIA 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 SACSEN and SPIAEN.



SPIA Block Diagram

SPIA Registers

There are three internal registers which control the overall operation of the SPIA interface. These are the SPIAD data register and two registers, SPIAC0 and SPIAC1.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SPIAC0	SASPI2	SASPI1	SASPI0	—	—	—	SPIAEN	SPIAICF
SPIAC1	—	—	SACKPOLB	SACKEG	SAMLS	SACSEN	SAWCOL	SATRF
SPIAD	D7	D6	D5	D4	D3	D2	D1	D0

SPIA Register List

SPIA Data Register

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

- **SPIAD 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**: SPIA data register bit 7 ~ bit 0

SPIA Control Registers

There are also two control registers for the SPIA interface, SPIAC0 and SPIAC1. The SPIAC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SPIAC1 register is used for other control functions such as LSB/MSB selection, write collision flag etc.

- **SPIAC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SASPI2	SASPI1	SASPI0	—	—	—	SPIAEN	SPIAICF
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	1	1	1	—	—	—	0	0

Bit 7~5 **SASPI2~SASPI0**: SPIA Operating Mode Control
000: SPIA master mode; SPIA clock is $f_{SYS}/4$
001: SPIA master mode; SPIA clock is $f_{SYS}/16$
010: SPIA master mode; SPIA clock is $f_{SYS}/64$
011: SPIA master mode; SPIA clock is f_{SUB}
100: SPIA master mode; SPIA clock is PTM1 CCRP match frequency/2
101: SPIA slave mode
110: Unimplemented
111: Unimplemented
These bits are used to control the SPIA Master/Slave selection and the SPIA Master clock frequency. The SPIA clock is a function of the system clock but can also be chosen to be sourced from PTM1 and f_{SUB}. If the SPIA Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4~2 Unimplemented, read as "0"

Bit 1 **SPIAEN**: SPIA Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SPIA interface. When the SPIAEN bit is cleared to zero to disable the SPIA interface, the SDIA, SDOA, SCKA and $\overline{SCSA}$ lines will lose their SPIA function and the SPIA operating current will be reduced to a minimum value. When the bit is high the SPIA interface is enabled.

Bit 0 **SPIAICF**: SPIA Incomplete Flag
0: SPIA incomplete condition is not occurred
1: SPIA incomplete condition is occured
This bit is only available when the SPIA is configured to operate in an SPIA slave mode. If the SPIA operates in the slave mode with the SPIAEN and SACSEN bits both being set high but the $\overline{SCSA}$ line is pulled high by the external master device before the

SPIA data transfer is completely finished, the SPIAICF bit will be set high together with the SATRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the SATRF bit will not be set high if the SPIAICF bit is set high by software application program.

- **SPIAC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SACKPOLB	SACKEG	SAMLS	SACSEN	SAWCOL	SATRF
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 **SACKPOLB**: SPIA clock line base condition selection
0: The SCKA line will be high when the clock is inactive
1: The SCKA line will be low when the clock is inactive
The SACKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCKA line will be low when the clock is inactive. When the SACKPOLB bit is low, then the SCKA line will be high when the clock is inactive.

Bit 4 **SACKEG**: SPIA SCKA clock active edge type selection
SACKPOLB=0
0: SCKA has high base level with data capture on SCKA rising edge
1: SCKA has high base level with data capture on SCKA falling edge
SACKPOLB=1
0: SCKA has low base level with data capture on SCKA falling edge
1: SCKA has low base level with data capture on SCKA rising edge
The SACKEG and SACKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPIA bus. These two bits must be configured before a data transfer is executed otherwise an erroneous clock edge may be generated. The SACKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCKA line will be low when the clock is inactive. When the SACKPOLB bit is low, then the SCKA line will be high when the clock is inactive. The SACKEG bit determines active clock edge type which depends upon the condition of the SACKPOLB bit.

Bit 3 **SAMLS**: SPIA 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 **SACSEN**: SPIA $\overline{\text{SCSA}}$ pin control
0: Disable
1: Enable
The SACSEN bit is used as an enable/disable for the $\overline{\text{SCSA}}$ pin. If this bit is low, then the $\overline{\text{SCSA}}$ pin will be disabled and placed into a floating condition. If the bit is high the $\overline{\text{SCSA}}$ pin will be enabled and used as a select pin.

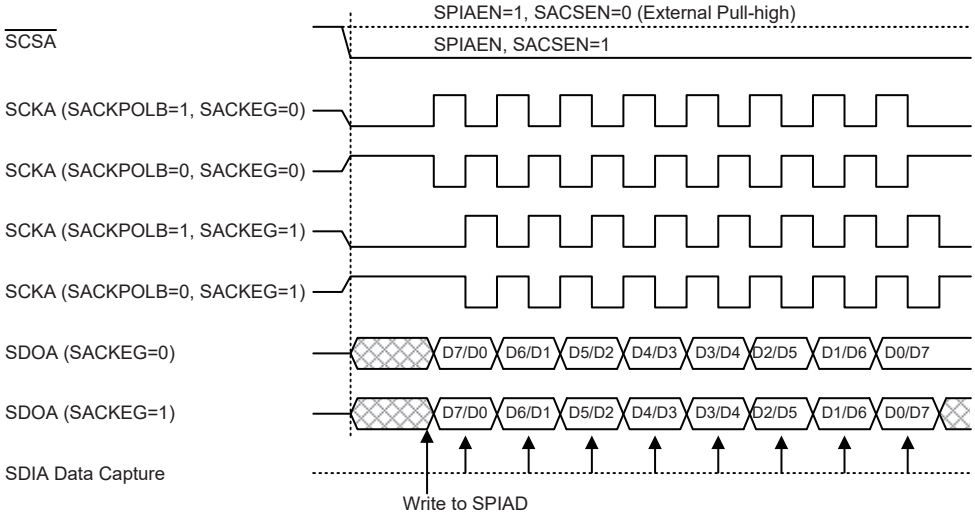
Bit 1 **SAWCOL**: SPIA write collision flag
0: No collision
1: Collision
The SAWCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SPIAD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared to zero by the application program.

Bit 0 **SATRF**: SPIA Transmit/Receive complete flag
0: SPIA data is being transferred
1: SPIA data transmission is completed
The SATRF bit is the Transmit/Receive Complete flag and is set "1" automatically when an SPIA data transmission is completed, but must set to zero by the application program. It can be used to generate an interrupt.

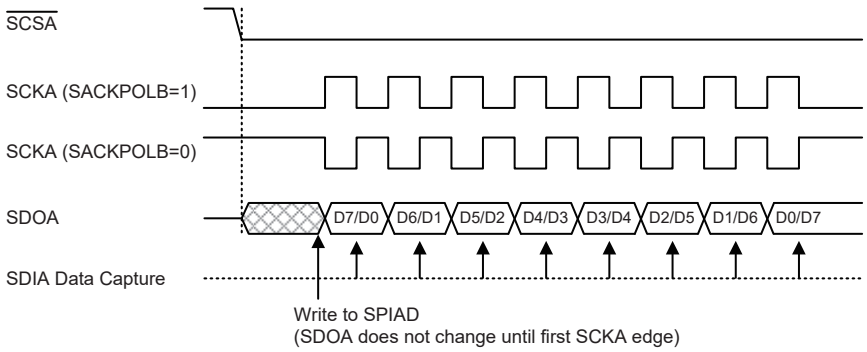
SPIA Communication

After the SPIA interface is enabled by setting the SPIAEN bit high, then in the Master Mode, when data is written to the SPIAD register, transmission/reception will begin simultaneously. When the data transfer is complete, the SATRF 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 SPIAD register will be transmitted and any data on the SDIA pin will be shifted into the SPIAD register. The master should output an $\overline{SCSA}$ 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 $\overline{SCSA}$ signal depending upon the configurations of the SACKPOLB bit and SACKEG bit. The accompanying timing diagram shows the relationship between the slave data and $\overline{SCSA}$ signal for various configurations of the SACKPOLB and SACKEG bits.

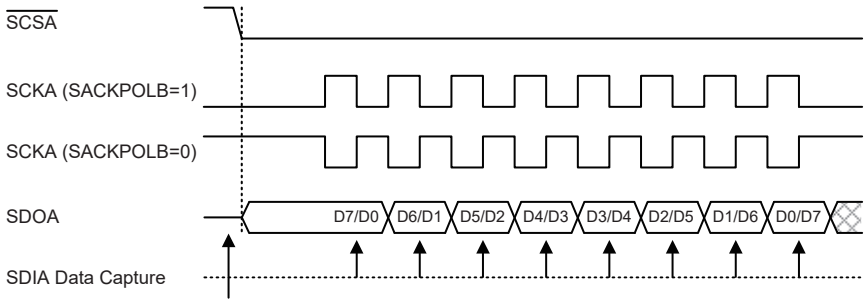
The SPIA will continue to function in certain IDLE Modes if the clock source used by the SPIA interface is still active.



SPIA Master Mode Timing

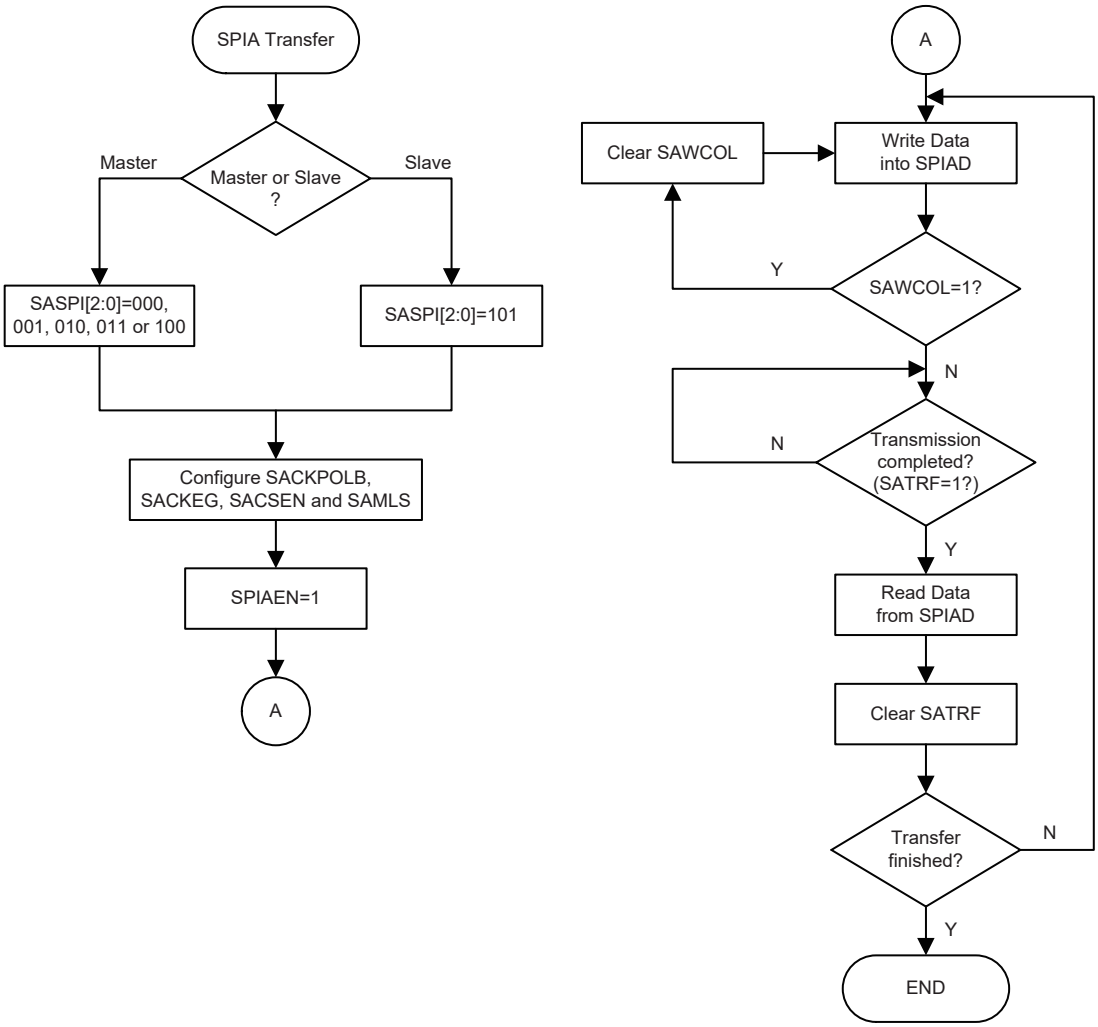


SPIA Slave Mode Timing – SACKEG=0



Note: For SPIA slave mode, if SPIAEN=1 and SACSEN=0, SPIA is always enabled and ignores the SCSA level.

SPIA Slave Mode Timing – SACKEG=1



SPIA Transfer Control Flowchart

SPIA Bus Enable/Disable

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

When the SPIA bus is disabled, SCKA, SDIA, SDOA, $\overline{\text{SCSA}}$can become I/O pins or other pin-shared functions using the corresponding pin-shared control bits.

SPIA Operation Steps

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

The SACSEN bit in the SPIAC1 register controls the overall function of the SPIA interface. Setting this bit high will enable the SPIA interface by allowing the $\overline{\text{SCSA}}$ line to be active, which can then be used to control the SPIA interface. If the SACSEN bit is low, the SPIA interface will be disabled and the $\overline{\text{SCSA}}$ line will be in a floating condition and can therefore not be used for control of the SPIA interface. If the SACSEN bit and the SPIAEN bit in the SPIAC0 register are set high, this will place the SDIA line in a floating condition and the SDOA line high. If in Master Mode the SCKA line will be either high or low depending upon the clock polarity selection bit SACKPOLB in the SPIAC1 register. If in Slave Mode the SCKA line will be in a floating condition. If SPIAEN is low then the bus will be disabled and $\overline{\text{SCSA}}$, SDIA, SDOA and SCKA will all become I/O pins or the other functions using the corresponding pin-shared control bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SPIAD 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 clock source and Master mode using the SASPI2~SASPI0 bits in the SPIAC0 control register.
- Step 2
  Setup the SACSEN bit and setup the SAMLS bit to choose if the data is MSB or LSB first, this must be same as the Slave device.
- Step 3
  Setup the SPIAEN bit in the SPIAC0 control register to enable the SPIA interface.
- Step 4
  For write operations: write the data to the SPIAD register, which will actually place the data into the TXRX buffer. Then use the SCKA and $\overline{\text{SCSA}}$ lines to output the data. After this go to step 5.
  For read operations: the data transferred in on the SDIA line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPIAD register.
- Step 5
  Check the SAWCOL 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 SATRF bit or wait for a SPIA serial bus interrupt.
- Step 7
  Read data from the SPIAD register.

- Step 8
  Clear SATRF.
- Step 9
  Go to step 4.

Slave Mode:

- Step 1
  Select the SPIA Slave mode using the SASPI2~SASPI0 bits in the SPIAC0 control register.
- Step 2
  Setup the SACSEN bit and setup the SAMLS 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 SPIAEN bit in the SPIAC0 control register to enable the SPIA interface.
- Step 4
  For write operations: write the data to the SPIAD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCKA and $\overline{\text{SCSA}}$ signal. After this, go to step 5.
  For read operations: the data transferred in on the SDIA line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPIAD register.
- Step 5
  Check the SAWCOL 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 SATRF bit or wait for a SPIA serial bus interrupt.
- Step 7
  Read data from the SPIAD register.
- Step 8
  Clear SATRF.
- Step 9
  Go to step 4.

Error Detection

The SAWCOL bit in the SPIAC1 register is provided to indicate errors during data transfer. The bit is set by the SPIA 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 SPIAD register takes place during a data transfer operation and will prevent the write operation from continuing.

Low Voltage Detector – LVD

These devices have a Low Voltage Detector function, also known as LVD. This enabled 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

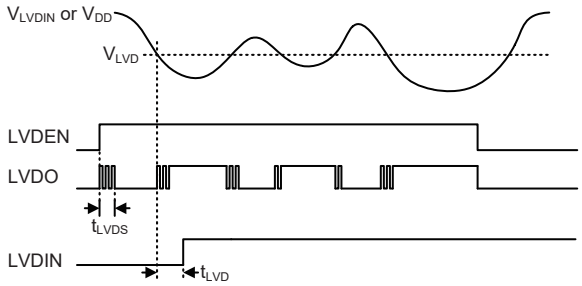
Bit 4 **LVDEN**: Low Voltage Detector Enable control
0: Disable
1: Enable

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 VLVD, there may be multiple bit LVDO transitions.



LVD Operation

The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. In this case, the 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.

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. These 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 or 1
USIM	USIME	USIMF	—
SPIA	SPIAE	SPIAF	—
Time Base	TBnE	TBnF	n=0 or 1
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~2
EEPROM	DEE	DEF	—
LVD	LVE	LVF	—
STM	STMPE	STMPF	—
	STMAE	STMAF	—
PTM	PTMnPE	PTMnPF	n=0~2
	PTMnAE	PTMnAF	

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	TB0F	MF2F	MF1F	MF0F	TB0E	MF2E	MF1E	MF0E
INTC2	—	SPIAF	USIMF	TB1F	—	SPIAE	USIME	TB1E
MFI0	PTM0AF	PTM0PF	STMAF	STMPF	PTM0AE	PTM0PE	STMAE	STMPE
MFI1	PTM2AF	PTM2PF	PTM1AF	PTM1PF	PTM2AE	PTM2PE	PTM1AE	PTM1PE
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**

Bit	7	6	5	4	3	2	1	0
Name	TB0F	MF2F	MF1F	MF0F	TB0E	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 **TB0F**: Time Base 0 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 **TB0E**: Time Base 0 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	—	SPIAF	USIMF	TB1F	—	SPIAE	USIME	TB1E
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 **SPIAF**: SPIA interrupt request flag
0: No request
1: Interrupt request

Bit 5 **USIMF**: USIM interrupt request flag
0: No request
1: Interrupt request

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

Bit 3 Unimplemented, read as "0"

Bit 2 **SPIAE**: SPIA interrupt control
0: Disable
1: Enable

Bit 1 **USIME**: USIM interrupt control
0: Disable
1: Enable

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

• **MFI0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTM0AF	PTM0PF	STMAF	STMPF	PTM0AE	PTM0PE	STMAE	STMPE
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 **PTM0AF**: PTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **PTM0PF**: PTM0 Comparator P match interrupt request flag
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 **PTM0AE**: PTM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **PTM0PE**: PTM0 Comparator P match interrupt control
0: Disable
1: Enable

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

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTM2AF	PTM2PF	PTM1AF	PTM1PF	PTM2AE	PTM2PE	PTM1AE	PTM1PE
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 **PTM2AF**: PTM2 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **PTM2PF**: PTM2 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 5 **PTM1AF**: PTM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTM1PF**: PTM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 **PTM2AE**: PTM2 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **PTM2PE**: PTM2 Comparator P match interrupt control
0: Disable
1: Enable

Bit 1 **PTM1AE**: PTM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTM1PE**: PTM1 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

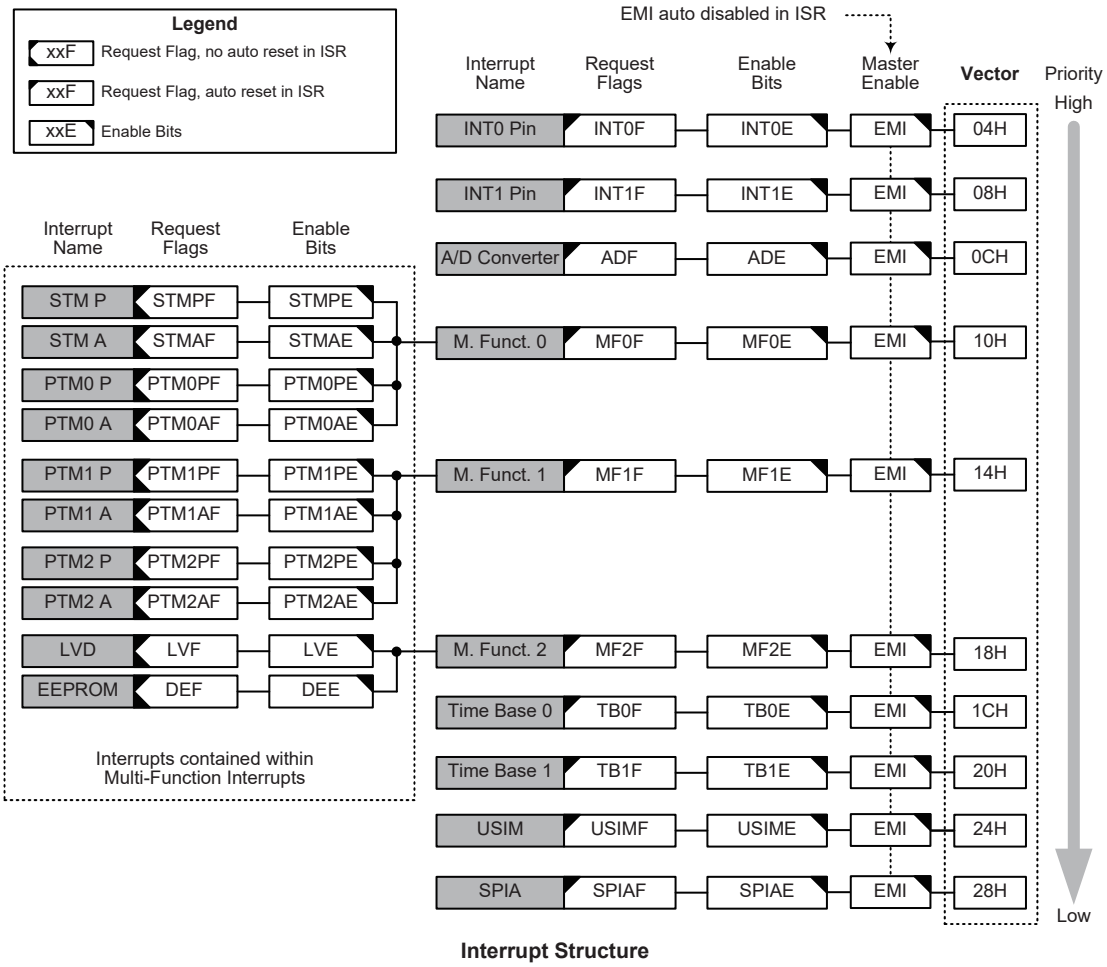
Interrupt Operation

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

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.

Universal Serial Interface Module Interrupt

The Universal Serial Interface Module Interrupt, also known as the USIM interrupt, will take place when the USIM Interrupt request flag, USIMF, is set. As the USIM interface can operate in three modes which are SPI mode, I^2C mode and UART mode, the USIMF flag can be set by different conditions depending on the selected interface mode.

If the SPI or I^2C mode is selected, the USIM interrupt can be triggered when a byte of data has been received or transmitted by the SPI/I^2C interface, or an I^2C slave address match occurs, or an I^2C bus time-out occurs. If the UART mode is selected, several individual UART conditions including a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up, can generate a USIM interrupt with the USIMF flag bit set high.

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, USIME, 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 Universal Serial Interface Interrupt flag, USIMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Note that if the USIM interrupt is triggered by the UART interface, after the interrupt has been serviced, the UUSR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

SPIA Interrupt

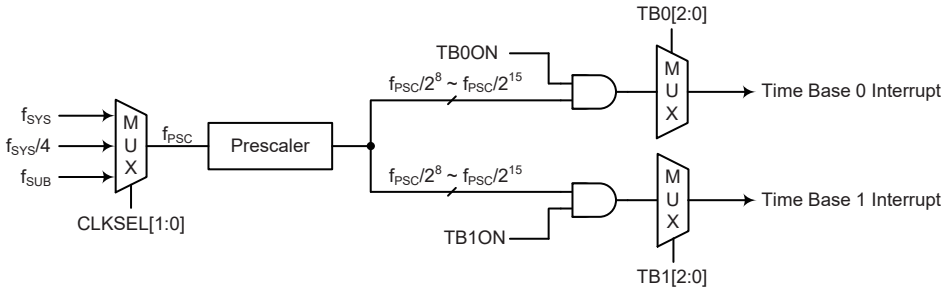
The Serial Peripheral Interface Interrupt, also known as the SPIA interrupt, will take place when the SPIA Interrupt request flag, SPIAF, is set, which occurs when a byte of data has been received or transmitted by the SPIA interface. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, SPIAE, 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, SPIAF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time 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: $2^{15}/f_{PSC}$

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.

Multi-function Interrupts

Within these devices there are up to three Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupts, 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. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

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

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

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, 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.

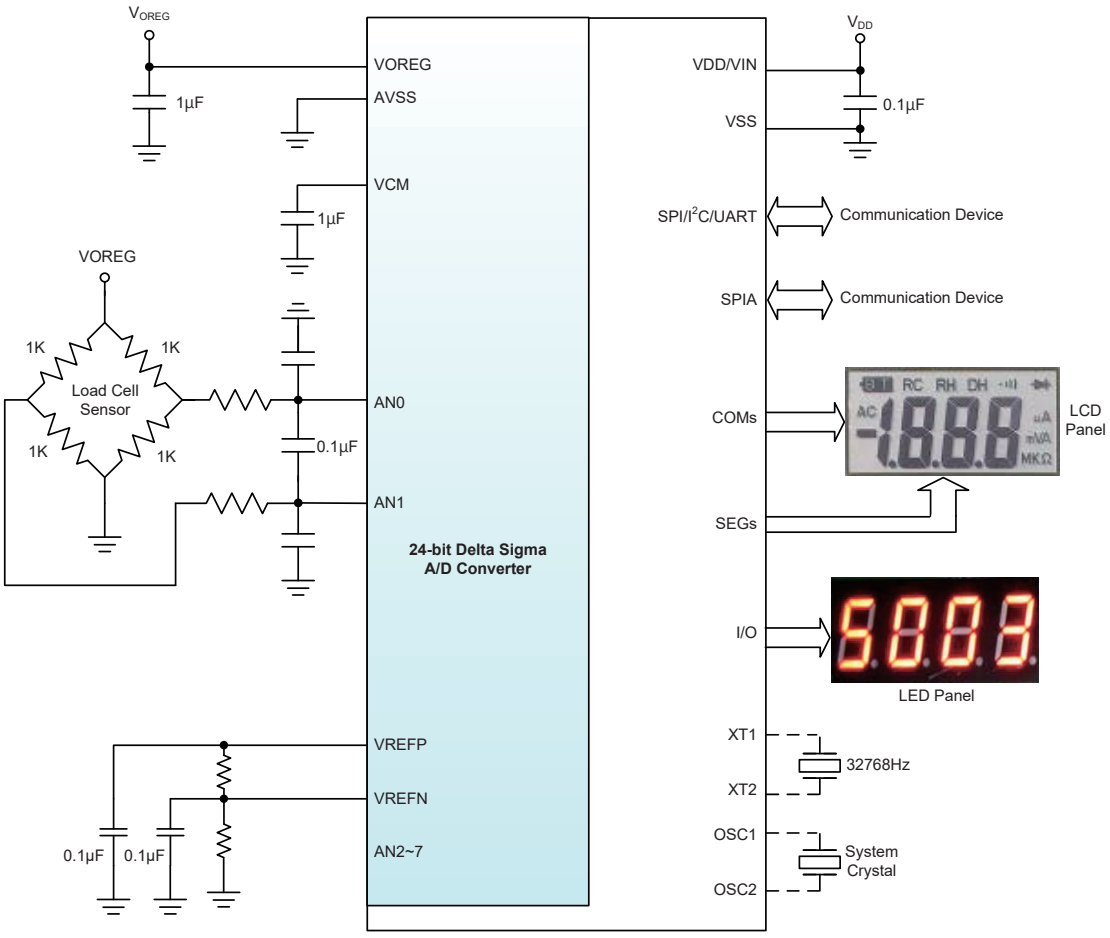
Configuration Options

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

No.	Options
Oscillator Option	
1	HIRC frequency selection: 1. 4MHz 2. 8MHz 3. 12MHz

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

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of several kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV, SC
ADDM A,[m]	Add ACC to Data Memory	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 $ACC \leftarrow ACC - [m] - \overline{C}$

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

SBC A, x Subtract immediate data from ACC with Carry

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation $[m] \leftarrow [m] - 1$
Skip if [m]=0

Affected flag(s) None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation $ACC \leftarrow [m] - 1$
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← 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

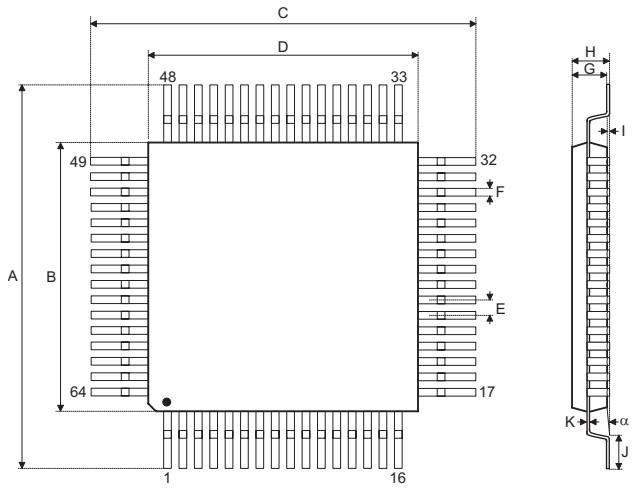
Package Information

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

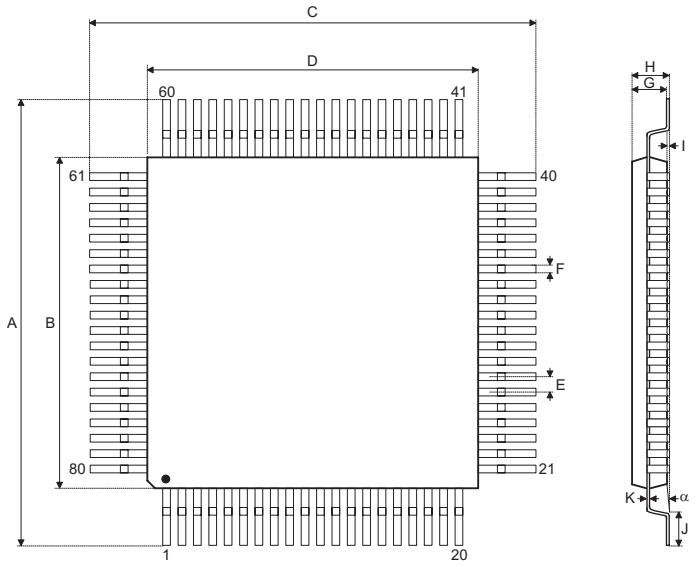
64-pin LQFP (7mm×7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.016 BSC	—
F	0.005	0.007	0.009
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.40 BSC	—
F	0.13	0.18	0.23
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

80-pin LQFP (10mm×10mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.472 BSC	—
B	—	0.394 BSC	—
C	—	0.472 BSC	—
D	—	0.394 BSC	—
E	—	0.015 BSC	—
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	12.00 BSC	—
B	—	10.00 BSC	—
C	—	12.00 BSC	—
D	—	10.00 BSC	—
E	—	0.40 BSC	—
F	0.13	0.18	0.23
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

Copyright© 2019 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at http://www.holtek.com.