TinyPower™ A/D Flash MCU with LCD & EEPROM

HT67F60A
HT67F70A

Revision: V1.30 Date: September 1, 2022

www.holtek.com

Table of Contents

Features **7**
CPU Features 7
Peripheral Features 7
General Description **8**
Selection Table **9**
Block Diagram **9**
Pin Assignment **10**
Pin Descriptions **13**
Absolute Maximum Ratings **19**
D.C. Characteristics **20**
A.C. Characteristics **22**
A/D Converter Characteristics **23**
LVD/LVR Electrical Characteristics **23**
Comparator Electrical Characteristics **24**
LCD Driver Electrical Characteristics **24**
Power-on Reset Characteristics **24**
System Architecture **25**
Clocking and Pipelining 25
Program Counter 26
Stack 27
Arithmetic and Logic Unit – ALU 27
Flash Program Memory **28**
Structure 28
Special Vectors 28
Look-up Table 29
Table Program Example 29
In Circuit Programming – ICP 30
On-Chip Debug Support – OCDS 31
In Application Programming – IAP 31
Data Memory **40**
Structure 40
Data Memory Addressing 41
General Purpose Data Memory 41
Special Purpose Data Memory 41

Special Function Register Description 43
Indirect Addressing Registers – IAR0, IAR1, IAR2 43
Memory Pointers – MP0, MP1H/MP1L, MP2H/MP2L 43
Bank Pointer – BP 45
Accumulator – ACC 45
Program Counter Low Register – PCL 46
Look-up Table Registers – TBLP, TBHP, TBLH 46
Status Register – STATUS 46

EEPROM Data Memory 48
EEPROM Data Memory Structure 48
EEPROM Registers 48
Reading Data from the EEPROM 50
Writing Data to the EEPROM 50
Write Protection 50
EEPROM Interrupt 50
Programming Considerations 51

Oscillator 52
Oscillator Overview 52
System Clock Configurations 52
External Crystal/Ceramic Oscillator – HXT 53
Internal High Speed RC Oscillator – HIRC 54
External 32.768 kHz Crystal Oscillator – LXT 54
Internal 32kHz Oscillator – LIRC 56
Supplementary Oscillators 56

Operating Modes and System Clocks 56
System Clocks 56
System Operation Modes 57
Control Registers 59
Fast Wake-up 61
Operating Mode Switching 62
Standby Current Considerations 66
Wake-up 67

Watchdog Timer 68
Watchdog Timer Clock Source 68
Watchdog Timer Control Register 68
Watchdog Timer Operation 69

Reset and Initialisation 70
Reset Functions 71
Reset Initial Conditions 75

Input/Output Ports 80
Pull-high Resistors 81
Port A Wake-up 81
I/O Port Control Registers 81
Pin-shared Functions 81
I/O Pin Structures 91
Programming Considerations 92
Timer Modules – TM 93
Introduction 93
TM Operation 93
TM Clock Source 94
TM Interrupts 94
TM External Pins 94
TM Input/Output Pin Control Register 95
Programming Considerations 96
Compact Type TM – CTM 97
Compact TM Operation 97
Compact Type TM Register Description 97
Compact Type TM Operation Modes 102
Standard Type TM – STM 108
Standard TM Operation 108
Standard Type TM Register Description 109
Standard Type TM Operation Modes 113
Enhanced Type TM – ETM 123
Enhanced TM Operation 123
Enhance Type TM Register Description 124
Enhanced Type TM Operation Modes 130
Analog to Digital Converter 147
A/D Overview 147
A/D Converter Register Description 148
A/D Operation 150
A/D Input Pins 151
Conversion Rate and Timing Diagram 152
Summary of A/D Conversion Steps 152
Programming Considerations 153
A/D Transfer Function 153
A/D Programming Examples 154
Comparators 156
Comparator Operation 156
Comparator Registers 156
Comparator interrupt 158
Programming Considerations 158

Serial Interface Module – SIM 159
SPI Interface 159
I²C Interface 165
Peripheral Clock Output 175
Peripheral Clock Operation 175
Peripheral Clock Registers 175
Serial Interface – SPIA 176
SPIA Interface Operation 176
SPIA Registers 177
SPIA Communication 180
SPIA Bus Enable/Disable 182
SPIA Operation 183
Error Detection 184
LCD Driver 185
LCD Memory 186
LCD Clock Source 187
LCD Registers 187
LCD Voltage Source Biasing 189
LCD Driver Output 190
Programming Considerations 193
Interrupts 194
Interrupt Registers 194
Interrupt Operation 205
External Interrupt 207
Comparator Interrupt 207
A/D Converter Interrupt 207
Time Base Interrupt 208
Multi-function Interrupt 210
Serial Interface Module Interrupt 210
SPIA Interface Interrupt 210
External Peripheral Interrupt 211
LVD Interrupt 211
EEPROM Interrupt 211
TM Interrupt 212
Interrupt Wake-up Function 212
Programming Considerations 212
Low Voltage Detector – LVD 213
LVD Register 213
LVD Operation 214
Configuration Options 215
Application Circuits 215

Instruction Set 216
Introduction 216
Instruction Timing 216
Moving and Transferring Data 216
Arithmetic Operations 216
Logical and Rotate Operation 217
Branches and Control Transfer 217
Bit Operations 217
Table Read Operations 217
Other Operations 217
Instruction Set Summary 218
Table Conventions 218
Extended Instruction Set 220
Instruction Definition 222
Extended Instruction Definition 231
Package Information 238
48-pin LQFP (7mm × 7mm) Outline Dimensions 239
64-pin LQFP (7mm × 7mm) Outline Dimensions 240
80-pin LQFP (10mm × 10mm) Outline Dimensions 241

Note that the HT67F70A device, although mentioned in this datasheet, has already been phased out and is presently no longer available.

Features

CPU Features

- Operating Voltage
  - f_{SYS}= 8MHz: 2.2V~5.5V
  - f_{SYS}= 12MHz: 2.7V~5.5V
  - f_{SYS}=20MHz: 4.5V~5.5V
- Up to 0.2μs instruction cycle with 20MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator Type
  - External Crystal – HXT
  - External 32.768kHz Crystal – LXT
  - High Speed Internal RC – HIRC
  - Low Speed Internal 32kHz RC – LIRC
- Fully integrated internal 4/8/12MHz oscillator requires no external components
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one to three instruction cycles
- Table read instructions
- 115 powerful instructions
- Up to 16-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Program Memory: 16K × 16 ~ 32K × 16
- Data Memory: 1024 × 8 ~ 2048 × 8
- True EEPROM Memory: 128 × 8
- In Application Programming function – IAP
- Watchdog Timer function
- Up to 47 bidirectional I/O lines
- Multiple pin-shared external interrupts
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
- Serial Interfaces Module – SIM for SPI or I^2C
- Single Serial Peripheral Interface – SPIA
- Dual Comparator functions
- Dual Time-Base functions for generation of fixed time interrupt signals
- 12-channel 12-bit resolution A/D converter
- Low voltage reset function
- Low voltage detect function
- Wide range of available package types

- Flash program memory can be re-programmed up to 10,000 times
- Flash program memory data retention > 10 years
- True EEPROM data memory can be re-programmed up to 100,000 times
- True EEPROM data memory data retention > 10 years

General Description

The series of devices are LCD type Flash Memory 8-bit high performance RISC architecture microcontroller which is designed for a wide range of applications. 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 12-bit A/D converter and dual comparator functions. 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 or I^2C interface functions, two popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments. A full choice of HXT, LXT, HIRC and LIRC oscillator functions are provided including a fully integrated system oscillator which requires no external components for its 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, Time-Base functions along with many other features ensure that the devices will find excellent use in applications such as electronic metering, environmental monitoring, handheld instruments, household appliances, electronically controlled tools, motor driving in addition to many others.

Selection Table

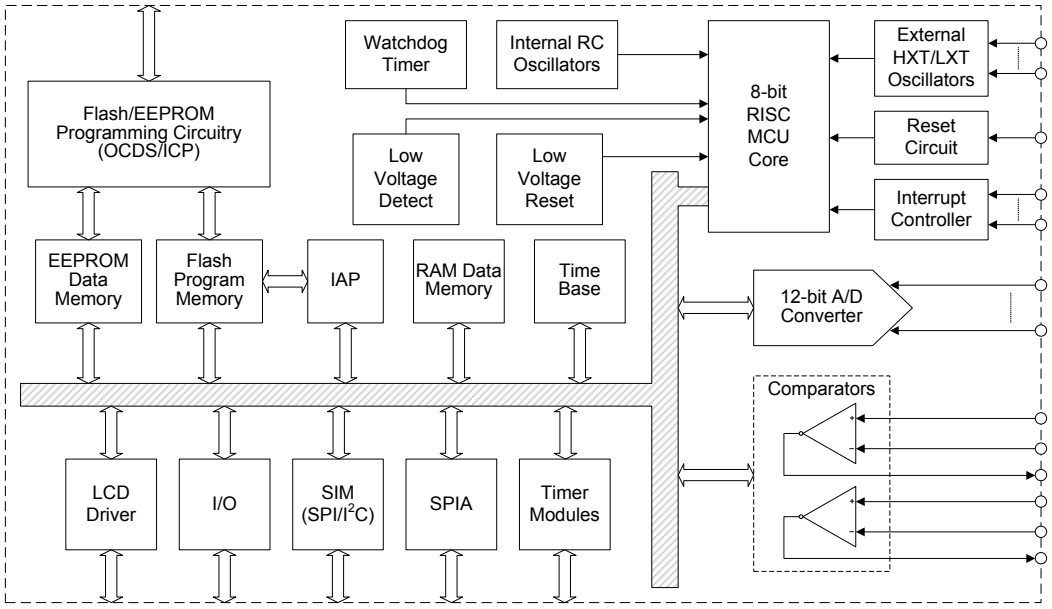
Most features are common to all devices. The main features distinguishing them are Program Memory and Data Memory capacity. The following table summarises the main features of each device.

Part No.	Program Memory	Data Memory	Data EEPROM	I/O	External Interrupt	A/D Converter	Timer Module
HT67F60A	16k × 16	1024 × 8	128 × 8	47	4	12-bit × 12	10-bit CTM × 2 16-bit STM × 3 10-bit ETM × 1
HT67F70A	32k × 16	2048 × 8	128 × 8	47	4	12-bit × 12	10-bit CTM × 2 16-bit STM × 3 10-bit ETM × 1

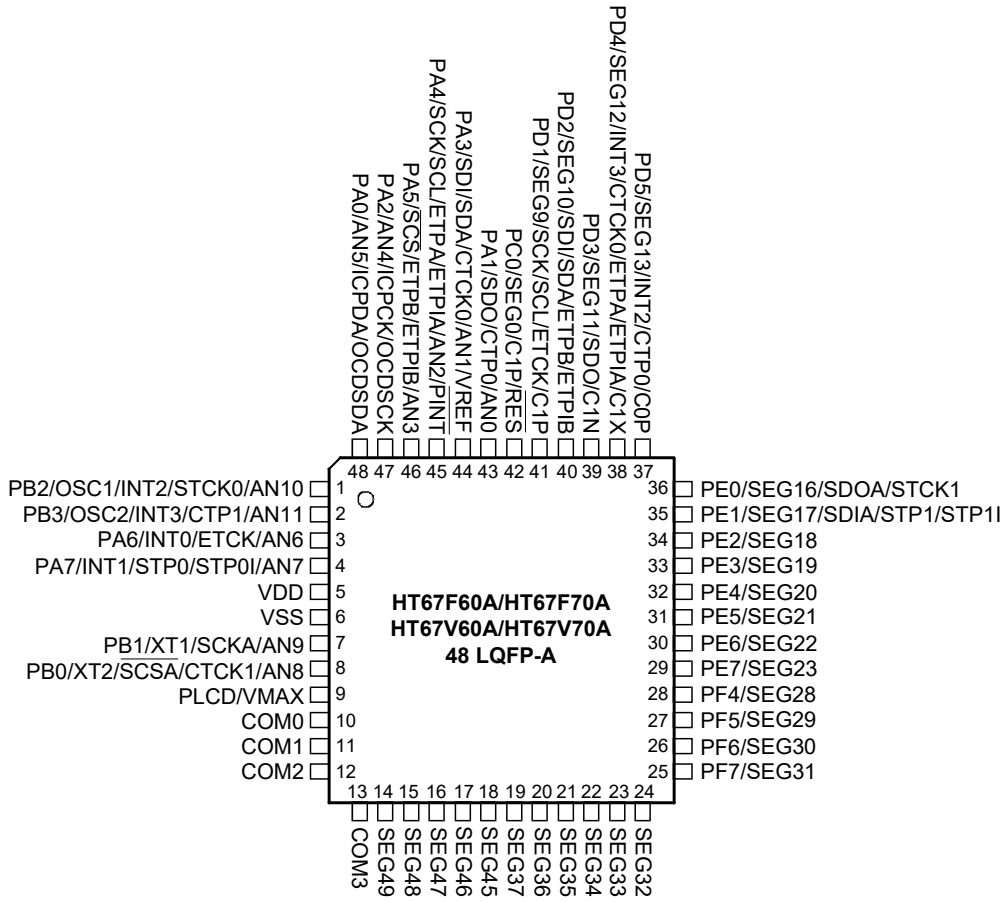
Part No.	SIM	SPIA	Time Base	Comp.	LCD Driver	Stacks	Package
HT67F60A	√	√	2	2	56 × 4	16	48/64/80 LQFP
HT67F70A	√	√	2	2	56 × 4	16	48/64/80 LQFP

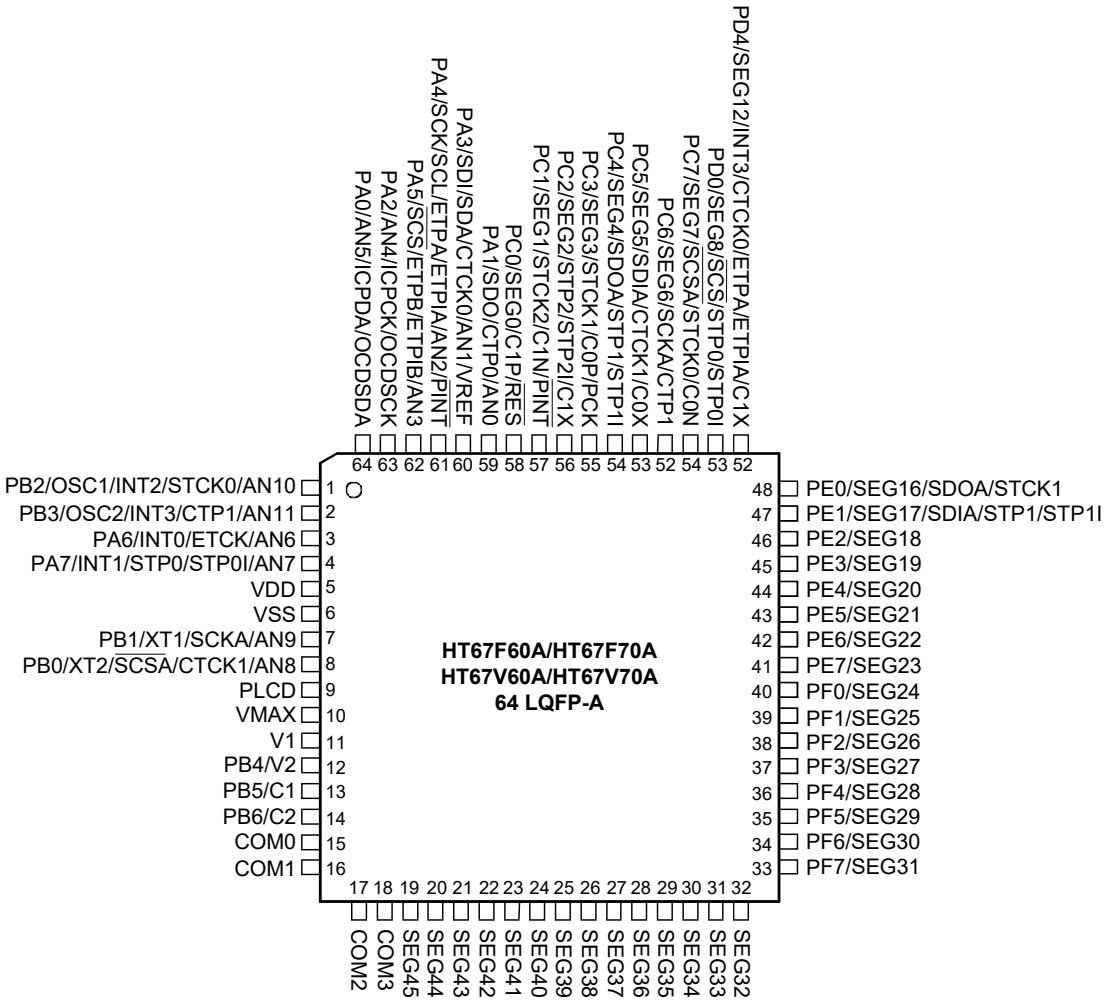
Note: As devices exist in more than one package format, the table reflects the situation for the package with the most pins.

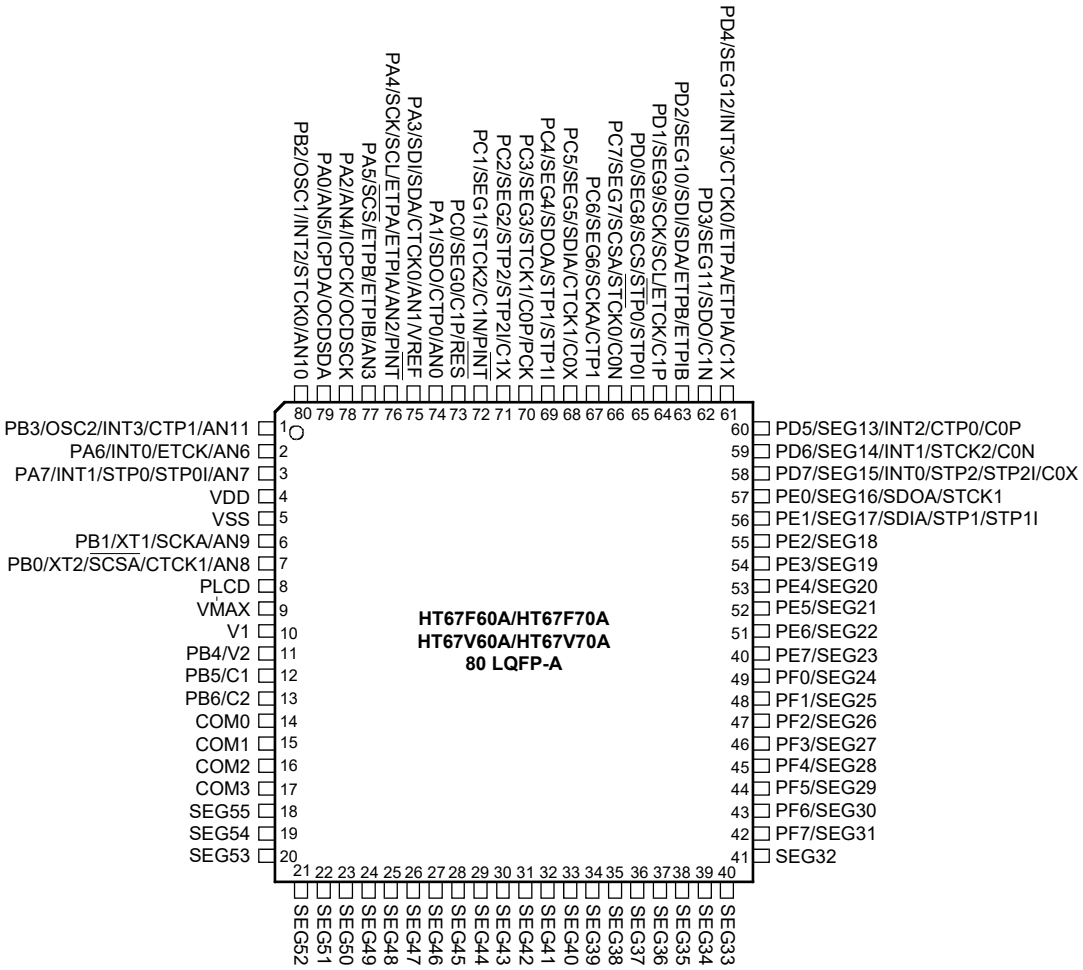
Block Diagram



Pin Assignment




PA0/AN5/ICPDA/OCDSDA 64
PA2/AN4/ICPCK/OCDSCK 63
PA5/SCS/ETPB/ETPIB/AN3 62
PA4/SCK/SCL/ETPA/ETPIA/AN2/PINT 61
PA3/SDI/SDA/CTCK0/AN1/VREF 60
PA1/SDO/CTP0/AN0 59
PC0/SEG0/C1P/RES 58
PC1/SEG1/STCK2/C1N/PINT 57
PC2/SEG2/STP2/STP2I/C1X 56
PC3/SEG3/STCK1/C0P/PCK 55
PC4/SEG4/SDOA/STP1/STP1I 54
PC5/SEG5/SDIA/CTCK1/C0X 53
PC6/SEG6/SCKA/CTP1 52
PC7/SEG7/SCSA/STCK0/C0N 54
PD0/SEG8/SCS/STP0/STP0I 53
PD4/SEG12/INT3/CTCK0/ETPA/ETPIA/C1X 52
PB2/OSC1/INT2/STCK0/AN10 1
PB3/OSC2/INT3/CTP1/AN11 2
PA6/INT0/ETCK/AN6 3
PA7/INT1/STP0/STP0I/AN7 4
VDD 5
VSS 6
PB1/XT1/SCKA/AN9 7
PB0/XT2/SCSA/CTCK1/AN8 8
PLCD 9
VMAX 10
V1 11
PB4/V2 12
PB5/C1 13
PB6/C2 14
COM0 15
COM1 16
HT67F60A/HT67F70A
HT67V60A/HT67V70A
64 LQFP-A
48 PE0/SEG16/SDOA/STCK1
47 PE1/SEG17/SDIA/STP1/STP1I
46 PE2/SEG18
45 PE3/SEG19
44 PE4/SEG20
43 PE5/SEG21
42 PE6/SEG22
41 PE7/SEG23
40 PF0/SEG24
39 PF1/SEG25
38 PF2/SEG26
37 PF3/SEG27
36 PF4/SEG28
35 PF5/SEG29
34 PF6/SEG30
33 PF7/SEG31
17 COM2
18 COM3
19 SEG45
20 SEG44
21 SEG43
22 SEG42
23 SEG41
24 SEG40
25 SEG39
26 SEG38
27 SEG37
28 SEG36
29 SEG35
30 SEG34
31 SEG33
32 SEG32



Note: 1. For 48 LQFP package type, only the R type bias LCD driver can be used.

2. If the pin-shared pin functions have multiple outputs simultaneously, the desired pin-shared function is determined by the corresponding software control bits except the functions determined by the configuration options.

3. The OCDSDA and OCDSCK pins are the OCDS dedicated pins and only available for the HT67Vx0A device. The HT67Vx0A device is the OCDS EV chip of the HT67Fx0A series of devices. It supports the "On-Chip Debug" function for debugging during development using the OCDSDA and OCDSCK pins connected to the Holtek HT_IDE development tools.

Pin Descriptions

With the exception of the power pins and some relevant transformer control pins, all pins on these devices can be referenced by their Port name, e.g. PA.0, PA.1 etc, which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Analog to Digital Converter, Timer Module pins etc. 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.

Pad Name	Function	OPT	I/T	O/T	Description
PA0/AN5/ICPDA/OCDSDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN5	PAS0	AN	—	A/D Converter analog input
	ICPDA	—	ST	CMOS	ICP Data/Address
	OCDSDA	—	ST	CMOS	OCDS Data/Address, for EV chip only.
PA1/SDO/CTP0/AN0	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDO	PAS0	—	CMOS	SPI data output
	CTP0	PAS0	—	CMOS	CTM0 output
	AN0	PAS0	AN	—	A/D Converter analog input
PA2/AN4/ICPCK/OCDSCK	PA2	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN4	PAS0	AN	—	A/D Converter analog input
	ICPCK	—	ST	CMOS	ICP Clock pin
	OCDSCK	—	ST	—	OCDS Clock pin, for EV chip only.
PA3/SDI/SDA/CTCK0/AN1/VREF	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDI	PAS0 IFS3	ST	—	SPI data input
	SDA	PAS0 IFS3	ST	NMOS	I^2C data line
	CTCK0	PAS0 IFS1	ST	—	CTM0 input
	AN1	PAS0	AN	—	A/D Converter analog input
	VREF	PAS0	AN	—	A/D Converter reference input
PA4/SCK/SCL/ETPA/ETPIA/AN2/PINT	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCK	PAS1 IFS3	ST	CMOS	SPI serial clock
	SCL	PAS1 IFS3	ST	NMOS	I^2C clock line
	ETPA	PAS1	—	CMOS	ETM A output
	ETPIA	PAS1 IFS1	ST	—	ETM A input
	AN2	PAS1	AN	—	A/D Converter analog input
	$\overline{\text{PINT}}$	PAS1 IFS0	ST	—	Peripheral interrupt

Pad Name	Function	OPT	I/T	O/T	Description
PA5/$\overline{SCS}$/ETPB/ ETPIB/AN3	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{SCS}$	PAS1 IFS3	ST	CMOS	SPI slave select
	ETPB	PAS1	—	CMOS	ETM B output
	ETPIB	PAS1 IFS1	ST	—	ETM B input
	AN3	PAS1	AN	—	A/D Converter analog input
PA6/INT0/ETCK/ AN6	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT0	PAS1 INTEG INTC0 IFS0	ST	—	External Interrupt 0
	ETCK	PAS1 IFS1	ST	—	ETM input
	AN6	PAS1	AN	—	A/D Converter analog input
PA7/INT1/STP0/ STP0I/AN7	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	STP0	PAS1	—	CMOS	STM0 output
	STP0I	PAS1 IFS1	ST	—	STM0 input
	AN7	PAS1	AN	—	A/D Converter analog input
PB0/XT2/$\overline{SCSA}$/ CTCK1/AN8	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	XT2	CO	—	LXT	LXT oscillator pin
	$\overline{SCSA}$	PBS0 IFS3	ST	CMOS	SPIA slave select
	CTCK1	PBS0 IFS2	ST	—	CTM1 input
	AN8	PBS0	AN	—	A/D Converter analog input
PB1/XT1/SCKA/ AN9	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	XT1	CO	LXT	—	LXT oscillator pin
	SCKA	PBS0 IFS3	ST	CMOS	SPIA serial clock
	AN9	PBS0	AN	—	A/D Converter analog input
PB2/OSC1/INT2/ STCK0/AN10	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	OSC1	CO	HXT	—	HXT oscillator pin
	INT2	PBS0 INTEG INTC3 IFS0	ST	—	External Interrupt 2
	STCK0	PBS0 IFS1	ST	—	STM0 input
	AN10	PBS0	AN	—	A/D Converter analog input

Pad Name	Function	OPT	I/T	O/T	Description
PB3/OSC2/INT3/CTP1/AN11	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	OSC2	CO	—	HXT	HXT oscillator pin
	INT3	PBS0 INTEG INTC3 IFS0	ST	—	External Interrupt 3
	CTP1	PBS0	—	CMOS	CTM1 output
	AN11	PBS0	AN	—	A/D Converter analog input
PB4/V2	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	V2	PBS1	—	—	LCD voltage pump
PB5/C1	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	C1	PBS1	—	—	LCD voltage pump
PB6/C2	PB6	PBPU PBS3	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	C2	PBS1	—	—	LCD voltage pump
PC0/SEG0/C1P/$\overline{RES}$	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG0	PCS0	—	LCD	LCD segment output
	C1P	PCS0 IFS4	AN	—	Comparator 1 non-inverting input
	$\overline{RES}$	RSTC	ST	—	Reset pin
PC1/SEG1/STCK2/C1N/$\overline{PINT}$	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG1	PCS0	—	LCD	LCD segment output
	STCK2	PCS0 IFS2	ST	—	STM2 input
	C1N	PCS0 IFS4	AN	—	Comparator 1 inverting input
	$\overline{PINT}$	PCS0 IFS0	ST	—	Peripheral interrupt input
PC2/SEG2/STP2/STP2I/C1X	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG2	PCS0	—	LCD	LCD segment output
	STP2	PCS0	—	CMOS	STM2 output
	STP2I	PCS0 IFS2	ST	—	STM2 input
	C1X	PCS0	—	CMOS	Comparator 1 output
PC3/SEG3/STCK1/C0P/PCK	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG3	PCS0	—	LCD	LCD segment output
	STCK1	PCS0 IFS2	ST	—	STM1 input
	C0P	PCS0 IFS4	AN	—	Comparator 0 non-inverting input
	PCK	PCS0	—	CMOS	Peripheral clock output

Pad Name	Function	OPT	I/T	O/T	Description
PC4/SEG4/SDOA/STP1/STP1I	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG4	PCS1	—	LCD	LCD segment output
	SDOA	PCS1	ST	CMOS	SPIA serial data output
	STP1	PCS1	—	CMOS	STM1 output
	STP1I	PCS1 IFS2	ST	—	STM1 input
PC5/SEG5/SDIA/CTCK1/C0X	PC5	PCPU PCS2	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG5	PCS1	—	LCD	LCD segment output
	SDIA	PCS1 IFS3	ST	CMOS	SPIA serial data input
	CTCK1	PCS1 IFS2	ST	—	CTM1 input
	C0X	PCS1	—	CMOS	Comparator 0 output
PC6/SEG6/SCKA/CTP1	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG6	PCS1	—	LCD	LCD segment output
	SCKA	PCS1 IFS3	ST	CMOS	SPIA serial clock
	CTP1	PCS1	—	CMOS	CTM1 output
PC7/SEG7/$\overline{\text{SCSA}}$/STCK0/C0N	PC7	PCPU PCS3	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG7	PCS1	—	LCD	LCD segment output
	$\overline{\text{SCSA}}$	PCS1 IFS3	ST	CMOS	SPIA slave select
	STCK0	PCS1 IFS1	ST	—	STM0 input
	C0N	PCS1 IFS4	AN	—	Comparator 0 inverting input
PD0/SEG8/$\overline{\text{SCS}}$/STP0/STP0I	PD0	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG8	PDS0	—	LCD	LCD segment output
	$\overline{\text{SCS}}$	PDS0 IFS3	ST	CMOS	SPI slave select
	STP0	PDS0	—	CMOS	STM0 inverted output
	STP0I	PDS0 IFS1	ST	—	STM0 input
PD1/SEG9/SCK/SCL/ETCK/C1P	PD1	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG9	PDS0	—	LCD	LCD segment output
	SCK	PDS0 IFS3	ST	CMOS	SPI serial clock
	SCL	PDS0 IFS3	ST	NMOS	I²C clock line
	ETCK	PDS0 IFS1	ST	—	ETM input
	C1P	PDS0 IFS4	AN	—	Comparator 1 non-inverting input

Pad Name	Function	OPT	I/T	O/T	Description
PD2/SEG10/ SDI/SDA/ETPB/ ETPIB	PD2	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG10	PDS0	—	LCD	LCD segment output
	SDI	PDS0 IFS3	ST	—	SPI data input
	SDA	PDS0 IFS3	ST	NMOS	I^2C data line
	ETPB	PDS0	—	CMOS	ETM B output
	ETPIB	PDS0 IFS1	ST	—	ETM B input
PD3/SEG11/ SDO/C1N	PD3	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG11	PDS0	—	LCD	LCD segment output
	SDO	PDS0	—	CMOS	SPI data output
	C1N	PDS0 IFS4	AN	—	Comparator 1 inverting input
PD4/SEG12/ INT3/CTCK0/ ETPA/ETPIA/C1X	PD4	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG12	PDS1	—	LCD	LCD segment output
	INT3	PDS1 INTEG INTC3 IFS0	ST	—	External Interrupt 3
	CTCK0	PDS1 IFS1	ST	—	CTM0 input
	ETPA	PDS1	—	CMOS	ETM A output
	ETPIA	PDS1 IFS1	ST	—	ETM A input
	C1X	PDS1	—	CMOS	Comparator 1 output
PD5/SEG13/ INT2/CTP0/C0P	PD5	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG13	PDS1	—	LCD	LCD segment output
	INT2	PDS1 INTEG INTC3 IFS0	ST	—	External Interrupt 2
	CTP0	PDS1	—	CMOS	CTM0 output
	C0P	PDS1 IFS4	AN	—	Comparator 0 non-inverting input
PD6/SEG14/ INT3/CTCK1/ C0N	PD6	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG14	PDS1	—	LCD	LCD segment output
	INT3	PDS1 INTEG INTC3 IFS0	ST	—	External Interrupt 3
	CTCK1	PDS1 IFS2	ST	—	CTM1 input
	C0N	PDS1 IFS4	AN	—	Comparator 0 inverting input

Pad Name	Function	OPT	I/T	O/T	Description
PD7/SEG15/ INT0/STP2/ STP2I/C0X	PD7	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG15	PDS1	—	LCD	LCD segment output
	INT0	PDS1 INTEG INTC0 IFS0	ST	—	External Interrupt 0
	STP2	PDS1	—	CMOS	STM2 output
	STP2I	PDS1 IFS2	ST	—	STM2 input
	C0X	PDS1	—	CMOS	Comparator 0 output
PE0/SEG16/ SDOA/STCK1	PE0	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG16	PES0	—	LCD	LCD segment output
	SDOA	PES0	ST	CMOS	SPIA serial data output
	STCK1	PES0 IFS2	ST	—	STM1 input
PE1/SEG17/ SDIA/STP1/ STP1I	PE1	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG17	PES0	—	LCD	LCD segment output
	SDIA	IFS3	ST	CMOS	SPI serial data input
	STP1	PES0	—	CMOS	STM1 output
	STP1I	PES0 IFS2	ST	—	STM1 input
PE2/SEG18	PE2	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG18	PES0	—	LCD	LCD segment output
PE3/SEG19	PE3	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG19	PES0	—	LCD	LCD segment output
PE4/SEG20	PE4	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG20	PES1	—	LCD	LCD segment output
PE5/SEG21	PE5	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG21	PES1	—	LCD	LCD segment output
PE6/SEG22	PE6	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG22	PES1	—	LCD	LCD segment output
PE7/SEG23	PE7	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG23	PES1	—	LCD	LCD segment output
PF0/SEG24	PF0	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG24	PFS0	—	LCD	LCD segment output
PF1/SEG25	PF1	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG25	PFS0	—	LCD	LCD segment output
PF2/SEG26	PF2	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG26	PFS0	—	LCD	LCD segment output

Pad Name	Function	OPT	I/T	O/T	Description
PF3/SEG27	PF3	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG27	PFS0	—	LCD	LCD segment output
PF4/SEG28	PF4	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG28	PFS1	—	LCD	LCD segment output
PF5/SEG29	PF5	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG29	PFS1	—	LCD	LCD segment output
PF6/SEG30	PF6	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG30	PFS1	—	LCD	LCD segment output
PF7/SEG31	PF7	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG31	PFS1	—	LCD	LCD segment output
SEG32~SEG55	SEGn	—	—	LCD	LCD segment output
COM0~COM3	COMn	—	—	LCD	LCD common output
V1	V1	—	—	—	LCD voltage pump
VMAX	VMAX	—	PWR	—	IC maximum voltage, connected to VDD, PLCD or V1.
PLCD	PLCD	—	PWR	—	LCD power supply
VDD	VDD	—	PWR	—	Positive Power supply.
VSS	VSS	—	PWR	—	Negative Power supply. Ground.

Note: OPT: Optional by configuration option (CO) or register option;
I/T: Input type; O/T: Output type;
PWR: Power; CO: Configuration option;
ST: Schmitt Trigger input; AN: Analog input;
CMOS: CMOS output; NMOS: NMOS output; LCD: LCD SEG/COM output;
HXT: High frequency crystal oscillator;
LXT: Low frequency crystal oscillator

Absolute Maximum Ratings

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

Note: These are stress ratings only. Stresses exceeding the range specified under “Absolute Maximum Ratings” may cause substantial damage to these devices. Functional operation of these 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

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (HXT)	—	$f_{SYS}=f_{HXT}$=8MHz	2.2	—	5.5	V
		—	$f_{SYS}=f_{HXT}$=12MHz	2.7	—	5.5	V
		—	$f_{SYS}=f_{HXT}$=20MHz	4.5	—	5.5	V
	Operating Voltage (HIRC)	—	$f_{SYS}=f_{HIRC}$=4MHz	2.2	—	5.5	V
		—	$f_{SYS}=f_{HIRC}$=8MHz	2.2	—	5.5	V
		—	$f_{SYS}=f_{HIRC}$=12MHz	2.7	—	5.5	V
I_{DD}	Operating Current (HXT)	3V	$f_{SYS}=f_{HXT}$=8MHz No load, all peripheral off	—	1.2	2.0	mA
		5V		—	2.8	4.5	mA
		3V	$f_{SYS}=f_{HXT}$=12MHz No load, all peripheral off	—	1.8	3.0	mA
		5V		—	4.0	6.0	mA
		5V	$f_{SYS}=f_{HXT}$=20MHz, no load, all peripheral off	—	5.5	8.5	mA
		3V	$f_{SYS}=f_{HXT}/2$, f_{HXT}=12MHz No load, all peripheral off	—	0.9	1.5	mA
		5V		—	2.1	3.3	mA
		3V	$f_{SYS}=f_{HXT}/4$, f_{HXT}=12MHz No load, all peripheral off	—	0.6	1.0	mA
		5V		—	1.6	2.5	mA
		3V	$f_{SYS}=f_{HXT}/8$, f_{HXT}=12MHz No load, all peripheral off	—	0.48	0.8	mA
		5V		—	1.2	2.0	mA
		3V	$f_{SYS}=f_{HXT}/16$, f_{HXT}=12MHz No load, all peripheral off	—	0.42	0.7	mA
		5V		—	1.1	1.7	mA
		3V	$f_{SYS}=f_{HXT}/32$, f_{HXT}=12MHz No load, all peripheral off	—	0.38	0.6	mA
		5V		—	1.0	1.5	mA
		3V	$f_{SYS}=f_{HXT}/64$, f_{HXT}=12MHz No load, all peripheral off	—	0.36	0.55	mA
		5V		—	1.0	1.5	mA
	Operating Current (HIRC)	3V	$f_{SYS}=f_{HIRC}$=4MHz No load, all peripheral off	—	0.7	1.2	mA
		5V		—	1.5	2.5	mA
		3V	$f_{SYS}=f_{HIRC}$=8MHz No load, all peripheral off	—	1.2	2.0	mA
		5V		—	2.8	4.5	mA
		3V	$f_{SYS}=f_{HIRC}$=12MHz No load, all peripheral off	—	1.5	3.0	mA
		5V		—	3.0	6.0	mA
	Operating Current (LXT)	3V	$f_{SYS}=f_{SUB}=f_{LXT}$=32.768kHz, LXTLP=0 No load, all peripheral off	—	10	20	μA
		5V		—	30	50	μA
		3V	$f_{SYS}=f_{SUB}=f_{LXT}$=32.768kHz, LXTLP=1 No load, all peripheral off	—	10	20	μA
		5V		—	30	50	μA
	Operating Current (LIRC)	3V	$f_{SYS}=f_{SUB}=f_{LIRC}$=32kHz No load, all peripheral off	—	10	20	μA
		5V		—	30	50	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB}	Standby Current (IDLE0)	3V	f_{SYS} off, f_{SUB} on, LXTLP=1 No load, all peripheral off	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
	Standby Current (IDLE1)	3V	f_{SYS}=12MHz on, f_{SUB} on No load, all peripheral off	—	0.6	1.0	mA
		5V		—	1.2	2.0	mA
		3V	f_{SYS}=12MHz/64 on, f_{SUB} on No load, all peripheral off	—	0.34	0.6	mA
		5V		—	0.85	1.2	mA
		3V	f_{SYS}=f_{LXT}=32.768kHz on, f_{SUB} on, LXTLP=1, No load, all peripheral off	—	1.9	4.0	μA
		5V		—	3.3	7.0	μA
	Standby Current (SLEEP0)	3V	f_{SYS} off, f_{SUB} off, WDT disabled No load, all peripheral off	—	0.1	1.0	μA
		5V		—	0.3	2.0	μA
	Stanby Current (SLEEP1)	3V	f_{SYS} off, f_{SUB}=f_{LIRC}=32kHz on, WDT enabled, No load, all peripheral off	—	1.3	5.0	μA
		5V		—	2.2	10.0	μA
		3V	f_{SYS} off, f_{SUB}=f_{LXT}=32.768Hz on, WDT enabled, No load, all peripheral off, LXTLP=0	—	1.3	5.0	μA
		5V		—	2.2	10.0	μA
		3V	f_{SYS} off, f_{SUB}=f_{LXT}=32.768Hz on, WDT enabled, No load, all peripheral off, LXTLP=1	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
V_{IL}	Input Low Voltage for I/O port (Except $\overline{RES}$ pin)	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	V
	Input Low Voltage ($\overline{RES}$ pin)	—	—	0	—	$0.4V_{DD}$	V
V_{IH}	Input High Voltage for I/O Ports (Except $\overline{RES}$ pin)	5V	—	3.5	—	5	V
		—	—	$0.8V_{DD}$	—	V_{DD}	V
	Input High Voltage ($\overline{RES}$ pin)	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	Sink Current for I/O Ports	3V	V_{OL}=$0.1V_{DD}$	4	8	—	mA
		5V	V_{OL}=$0.1V_{DD}$	10	20	—	mA
I_{OH}	Source Current for I/O Ports	3V	V_{OH}=$0.9V_{DD}$	-2	-4	—	mA
		5V	V_{OH}=$0.9V_{DD}$	-5	-10	—	mA
R_{PH}	Pull-high Resistance for I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (HXT)	2.2V~5.5V	—	—	8	—	MHz
		2.7V~5.5V		—	12	—	MHz
		4.5V~5.5V		—	20	—	MHz
	System Clock (HIRC)	2.2V~5.5V	—	—	4	—	MHz
				—	8	—	MHz
		2.7V~5.5V		—	12	—	MHz
	System Clock (LXT)	2.2V~5.5V	—	—	32.768	—	kHz
	System Clock (LIRC)	2.2V~5.5V	—	—	32	—	kHz
f_{HXT}	High Speed Crystal Oscillator Clock (HXT)	2.2V~5.5V	—	0.4	—	8	MHz
		2.7V~5.5V		0.4	—	12	MHz
		4.5V~5.5V		0.4	—	20	MHz
f_{HIRC}	High Speed RC Oscillator Clock (HIRC)	5V	Ta = 25°C	Typ. -2%	4	Typ. +2%	MHz
				Typ. -2%	8	Typ. +2%	MHz
				Typ. -2%	12	Typ. +2%	MHz
		3.0V~5.5V	Ta = 0°C ~ +85°C	Typ. -10%	4	Typ. +10%	MHz
				Typ. -10%	4	Typ. +10%	MHz
				Typ. -10%	4	Typ. +10%	MHz
f_{LXT}	Low Speed Crystal Oscillator Clock (LXT)	—	—	—	32.768	—	kHz
f_{LIRC}	Low Speed RCl Oscillator Clock (LIRC)	5V	Ta = 25°C	Typ. -3%	32	Typ. +3%	kHz
t_{TCK}	TM TCK Input Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TP}	TM TPI Input Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	External Interrupt Minimum Input Pulse Width	—	—	10	—	—	μs
t_{RES}	External Reset Minimum Low Pulse Width	—	—	10	—	—	μs
t_{SST}	System Start-up Time Period (Wake-up from Power down mode, f_{SYS} off)	—	f_{SYS}= f_{HXT}	1024	—	—	t_{HXT}
			f_{SYS}= f_{HIRC}	16	—	—	t_{HIRC}
			f_{SYS}= f_{LXT}	1024	—	—	t_{LXT}
			f_{SYS}= f_{LIRC}	2	—	—	t_{LIRC}
	System Start-up Time Period (Wake-up from Power down mode, f_{SYS} on)	—	—	2	—	—	t_{SYS}
	System Start-up Time Period (Mode Switching between NORMAL and SLOW)	—	f_{HXT} off → on	1024	—	—	t_{HXT}
		—	f_{HIRC} off → on	16	—	—	t_{HIRC}
		—	f_{LXT} off → on	1024	—	—	t_{LXT}
		—	f_{LIRC} off → on	2	—	—	t_{LIRC}
t_{RSTD}	System Reset Delay Time (Power On Reset, RSTC Software reset LVR hardware reset, LVRC software reset, WDTC software reset)	—	—	25	50	100	ms
	System Reset Delay Time ($\overline{RES}$ reset, WDT overflow reset)	—	—	8.3	16.7	33.3	ms
t_{EERD}	EEPROM Read Time	—	—	—	—	4	t_{SYS}
t_{EEWR}	EEPROM Write Timet	—	—	—	2	4	ms

Note: t_{SYS}= 1/f_{SYS}

A/D Converter Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.7	—	5.5	V
V_{ADI}	Input Voltage	—	—	0	—	V_{DD}/V_{REF}	V
V_{REF}	Reference Voltage	—	—	2	—	V_{DD}+0.1V	V
DNL	Differential Non-linearity	3V	$V_{REF}=V_{DD}$, t_{ADCK} =0.5μs	—	—	±3	LSB
		5V					
INL	Integral Non-linearity	3V	$V_{REF}=V_{DD}$, t_{ADCK} =0.5μs	—	—	±4	LSB
		5V					
I_{ADC}	Additional Current for A/D Converter enable	3V	No load, t_{ADCK} =0.5μs	—	1.0	2.0	mA
		5V		—	1.5	3.0	mA
t_{ADCK}	Clock Period	—	—	0.5	—	10	μs
t_{ADC}	Conversion Time (Including Sample and Hold Time)	—	—	—	16	—	t_{ADCK}
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs

LVD/LVR Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR Enable, voltage select 2.1V	Typ. - 5%	2.1	Typ. + 5%	V
			LVR Enable, voltage select 2.55V		2.55		
			LVR Enable, voltage select 3.15V		3.15		
			LVR Enable, voltage select 3.8V		3.8		
V_{LVD}	Low Voltage Detector Voltage	—	LVD Enable, voltage select 2.0V	Typ. - 5%	2.0	Typ. + 5%	V
			LVD Enable, voltage select 2.2V		2.2		
			LVD Enable, voltage select 2.4V		2.4		
			LVD Enable, voltage select 2.7V		2.7		
			LVD Enable, voltage select 3.0V		3.0		
			LVD Enable, voltage select 3.3V		3.3		
			LVD Enable, voltage select 3.6V		3.6		
			LVD Enable, voltage select 4.0V		4.0		
V_{BG}	Bandgap Reference Voltage	—	—	Typ. - 5%	1.04	Typ. + 5%	V
I_{OP}	LVD/LVR Operating Current	5V	LVD/LVR Enable, VBGEN=0	—	20	25	μA
		5V	LVD/LVR Enable, VBGEN=1	—	180	200	μA
t_{BGS}	V_{BG} Turn on Stable Time	—	No load	—	—	150	μs
t_{LVDS}	LVDO stable time	—	For LVR Enable, VBGEN=0, LVD Disable → Enable	—	—	15	μs
		—	For LVR Disable, VBGEN=0, LVD Disable → Enable	—	—	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

Comparator Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.7	—	5.5	V
I_{CMP}	Additional Current for Comparator enabled	3V	—	—	50	75	μA
		5V	—	—	85	130	μA
V_{OS}	Input Offset Voltage	5V	—	-10	—	+10	mV
V_{HYS}	Hesteresis	5V	—	20	40	60	mV
V_{CM}	Common Mode Voltage Range	—	—	0	—	V_{DD} -1.4	V
A_{OL}	Open Loop Gain	—	—	60	80	—	dB
t_{RP}	Response Time	3V/5V	With 100mV overdrive (Note)	—	—	2	μs

Note: Measured with comparator one input pin at V_{CM} = (V_{DD}-1.4)/2 while the other pin input transition from V_{SS} to (V_{CM} +100mV) or from V_{DD} to (V_{CM} -100mV).

LCD Driver Electrical Characteristics

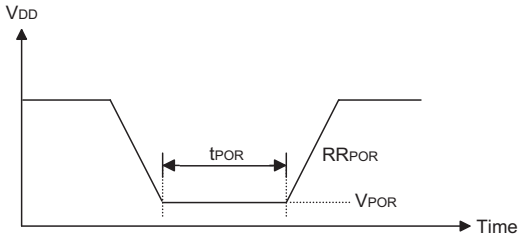
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OL}	Sink Current for LCD pins	3V	V_{PLCD}=3V, V_{OL}=0.1V_{PLCD}	210	420	—	mA
		5V	V_{PLCD}=5V, V_{OL}=0.1V_{PLCD}	350	700	—	mA
I_{OH}	Source Current for LCD pins	3V	V_{PLCD}=3V, V_{OL}=0.9V_{PLCD}	-80	-160	—	mA
		5V	V_{PLCD}=5V, V_{OL}=0.9V_{PLCD}	-180	-360	—	mA
I_{LCD}	LCD Operating Current (R type)	3V	No load, 1/3 bias, R_T=1170kΩ	—	—	5	μA
		5V		—	—	7.5	μA
		3V	No load, 1/3 bias, R_T=2250kΩ	—	—	23	μA
		5V		—	—	40	μA
		3V	No load, 1/3 bias, R_T=60kΩ	—	—	86	μA
		5V		—	—	145	μA
	LCD Operating Current (C type)	3V	No load, 1/3 bias	—	—	1	μA
		5V		—	—	2	μA

Power-on Reset Characteristics

Ta=25°C

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

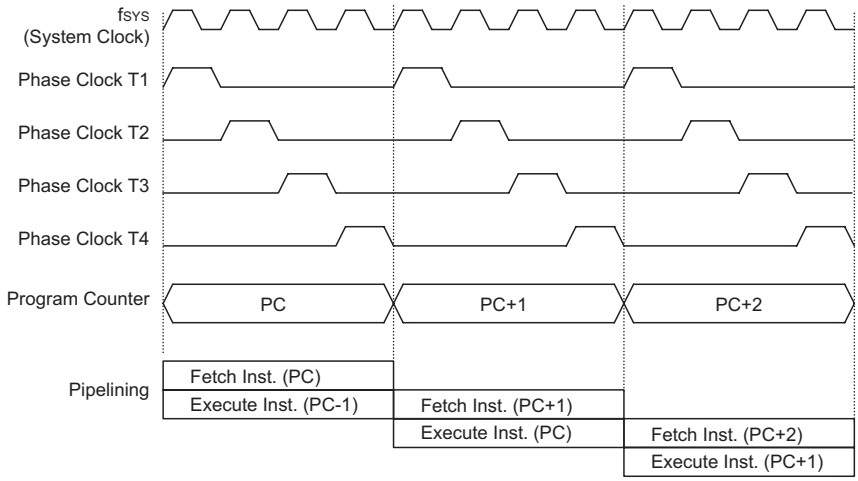


System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The range of 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 cycle, with the exception of branch or call instructions. 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

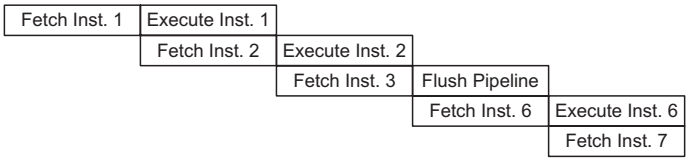
The main system clock, derived from either a HXT, LXT, HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry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

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



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demand a jump to a non-consecutive Program Memory address which is located in a certain program memory bank selected by the program memory bank pointer bits. 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	
	High Byte	Low Byte (PCL)
HT67F60A	BP0, PC12~PC8	PC7~PC0
HT67F60A	BP1~BP0, PC12~PC8	PC7~PC0

Program Counter

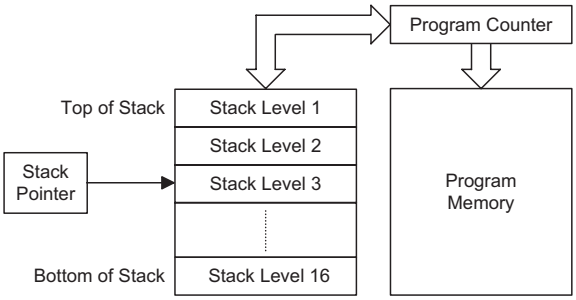
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

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

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

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



Arithmetic and Logic Unit – ALU

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

- Arithmetic operations:ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA, LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA, LAND, LOR, LXOR, LANDM, LORM, LXORM, LCPL, LCPLA
- Rotation: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC, LRRA, LRR, 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, LSZ, LSZA, LSNZ, LSIZ, LSDZ, LSIZA, LSDZA

Flash Program Memory

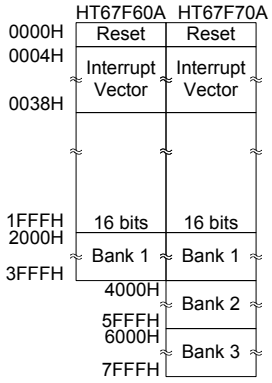
The Program Memory is the location where the user code or program is stored. For these devices series the Program Memory are Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the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

Device	Capacity	Banks
HT67F60A	16K × 16	0~1
HT67F70A	32K × 16	0~3

The series of devices have their Program Memory divided into two or four Banks, Bank 0~Bank 1 or Bank 0~Bank 3 respectively. The required Bank is selected using Bit 0 or Bit 0~1 of the BP Register dependent upon which device is selected.



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, the data can be retrieved from the program memory using the corresponding extended table read instruction such as "LTABRD [m]" or "LTABRDL [m] "respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

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

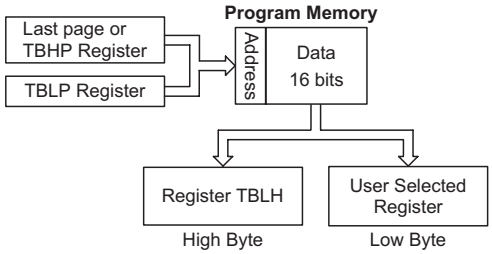


Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the last page which is stored there using the "ORG" and "rombank" statements. The value at this ORG statement is "1F00H" which is located in ROM Bank 3 which refers to the start address of the last page within the 8K words Program Memory in Bank 3 of the device with 32K words program memory. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "7F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

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

Table Read Program Example

```
rombank 3 code3
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,7fh              ; initialise high table pointer
mov tbhp,a             ; it is not necessary to set tbhp if executing tabrdl
:
tabrdc tempreg1
tabrdl tempreg1        ; transfers value in table referred by table pointer to tempreg1
                       ; data at program memory address 7F06H transferred to
                       ; tempreg1 and TBLH
dec tblp               ; reduce value of table pointer by one
tabrdc tempreg2
tabrdl tempreg2        ; transfers value in table referenced by table pointer to tempreg2
                       ; data at program memory address 7F05H 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
:
code3 .section 'code'
org 1F00h              ; sets initial address of lastpage
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
:
```

In Circuit Programming – ICP

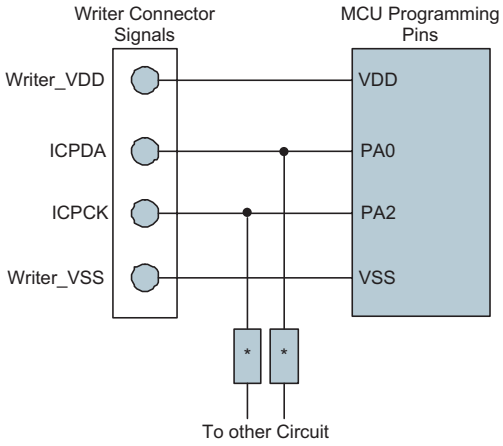
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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 and EEPROM data 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 are 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 1k or the capacitance of * must be less than 1nF.

On-Chip Debug Support – OCDS

There is an EV chip named HT67Vx0A which is used to emulate the real MCU device named HT67Fx0A. 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, HT67Vx0A and HT67Fx0A respectively, are almost functional compatible except the "On-Chip Debug" function and package types. 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 OCDS Pins	Pin Description
OCDSDA	OCDSDA	On-Chip Debug Support Data/Address input/output
OCDSCK	OCDSCK	On-Chip Debug Support Clock input
VDD	VDD	Power Supply
VSS	VSS	Ground

In Application Programming – IAP

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

- Erase Page: 64 words/page
- Writing: 64 words/time
- Reading: 1 word/time

In Application Program Control Registers

The Address register, FARL and FARH, the Data registers, FD0L/FD0H, FD1L/FD1H, FD2L/FD2H and FD3L/FD3H, and the Control registers, FC0, FC1 and FC2, are the corresponding Flash access registers located in Data Memory sector 1 for IAP. If using the indirect addressing method to access the FC0, FC1 and FC2 registers, all read and write operations to the registers must be performed using the Indirect Addressing Register, IAR1 or IAR2, and the Memory Pointer pair, MP1L/MP1H or MP2L/MP2H. Because the FC0, FC1 and FC2 control registers are located at the address of 43H~45H in Data Memory sector 1, the desired value ranged from 43H to 45H must first be written into the MP1L or MP2L Memory Pointer low byte and the value "01H" must also be written into the MP1H or MP2H Memory Pointer high byte.

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	A7	A6	A5	A4	A3	A2	A1	A0
FARH (HT67F60A)	—	—	A13	A12	A11	A10	A9	A8
FARH (HT67F70A)	—	A14	A13	A12	A11	A10	A9	A8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Registers List

FC0 Register

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

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

Bit 6~4 **FMOD2~FMOD0**: Mode selection
000: Write program memory
001: Page erase program memory
010: Reserved
011: Read program memory
10x: Reserved
110: FWEN mode – Flash memory Write function Enabled mode
111: Reserved

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

Bit 2 **FWT**: Flash memory Write Initiate control
0: Do not initiate Flash memory write or Flash memory write process is completed
1: Initiate Flash memory write process
This bit is set by software and cleared by hardware when the Flash memory write process is completed.

Bit 1 **FRDEN**: Flash memory Read Enable control
0: Flash memory read disable
1: Flash memory read enable

Bit 0 **FRD**: Flash memory Read Initiate control
0: Do not initiate Flash memory read or Flash memory read process is completed
1: Initiate Flash memory read process
This bit is set by software and cleared by hardware when the Flash memory read process is completed.

FC1 Register

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

Bit 7~0 **D7~D0**: Whole chip reset pattern
When user writes a specific value of "55H" to this register, it will generate a reset signal to reset 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 Write Buffer Clear process or Write Buffer Clear process is completed
1: Initiate Write Buffer Clear process
This bit is set by software and cleared by hardware when the Write Buffer Clear process is completed.

FARL Register

Bit	7	6	5	4	3	2	1	0
Name	A7	A6	A5	A4	A3	A2	A1	A0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

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

FARH Register – HT67F60A

Bit	7	6	5	4	3	2	1	0
Name	—	—	A13	A12	A11	A10	A9	A8
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 0

FARH Register – HT67F70A

Bit	7	6	5	4	3	2	1	0
Name	—	A14	A13	A12	A11	A10	A9	A8
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 0

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

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

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

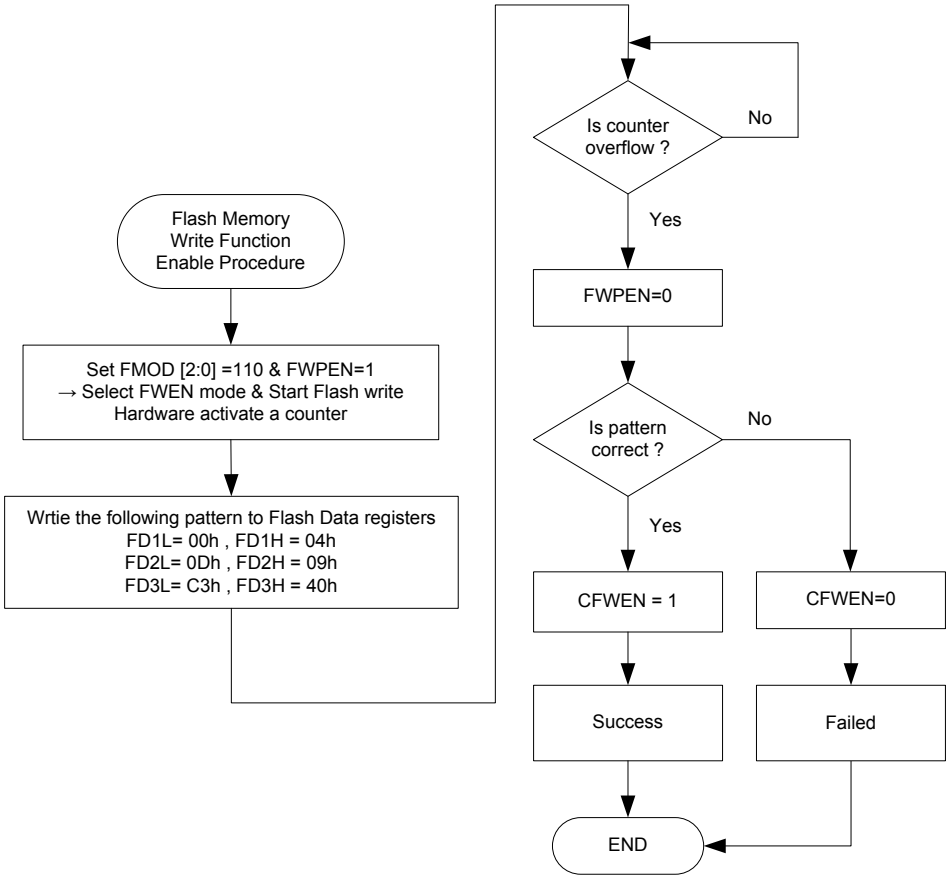
Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

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

Flash Memory Write Function Enable Procedure

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

1. Write "110" into the FMOD2~FMOD0 bits to select the FWEN mode.
2. Set the FWPEN bit to "1". The step 1 and step 2 can be executed simultaneously.
3. The pattern data with a sequence of 00H, 04H, 0DH, 09H, C3H and 40H must be written into the FD1L, FD1H, FD2L, FD2H, FD3L and FD3H registers respectively.
4. A counter with a time-out period of 300μs will be activated to allow users writing the correct pattern data into the FD1L/FD1H ~ FD3L/FD3H register pairs. The counter clock is derived from LIRC oscillator.
5. When the counter overflows, the FWPEN bit will automatically be cleared to 0 by hardware followed by pattern check.
6. If the pattern data is correct, the CFWEN bit will be set to 1 by hardware to indicate that the Flash memory write operation is successfully enabled.
7. Once the Flash memory write operation is enabled, the user can change the Flash ROM data through the Flash control register.
8. To disable the Flash memory write operation, the user can clear the CFWEN bit to 0.



Flash Memory Write Function Enable Procedure

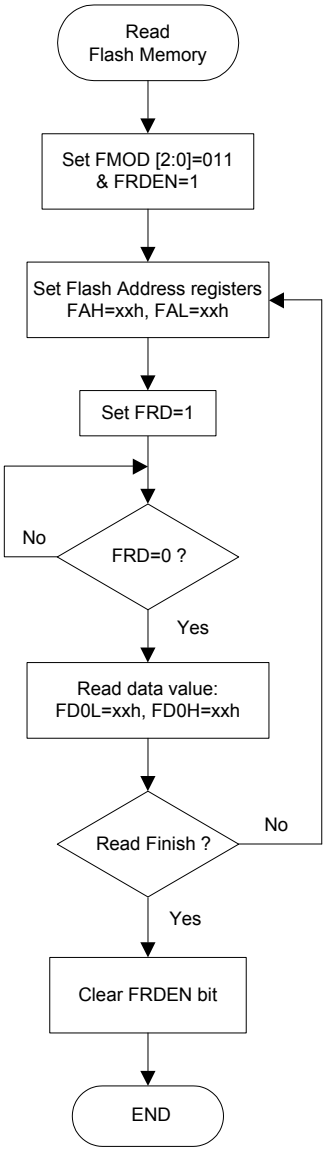
Flash Memory Write Procedure

After the Flash memory write function is successfully enabled through the preceding IAP procedure, users must first erase the corresponding Flash memory block and then initiate the Flash memory write operation. Since the number of the page erase operation is 64 words per page, the available page erase address is specified by FARH register and the content of bit 7 ~ bit 6 in the FARL register.

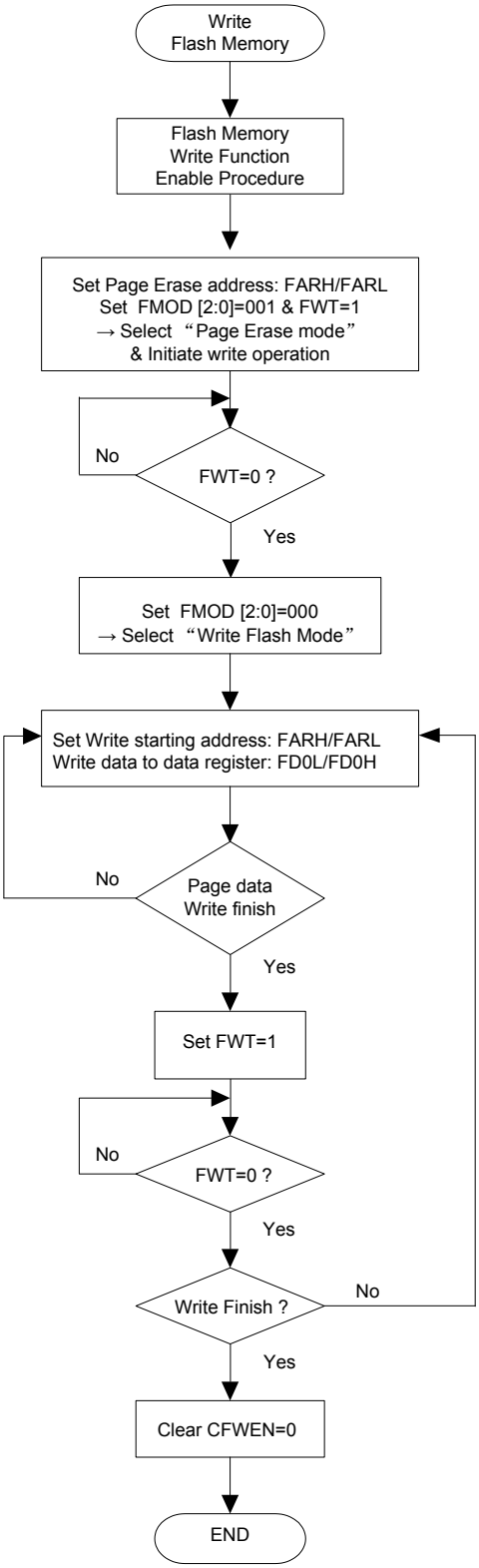
Erase Page	FARH	FARL [7:6]	FARL [5:0]
0	0000 0000	00	xxxx
1	0000 0000	01	xxxx
2	0000 0000	10	xxxx
3	0000 0000	11	xxxx
4	0000 0001	00	xxxx
5	0000 0001	01	xxxx
6	0000 0001	10	xxxx
7	0000 0001	11	xxxx
8	0000 0010	00	xxxx
9	0000 0010	01	xxxx
:	:	:	:
252	0011 1111	00	xxxx
253	0011 1111	01	xxxx
254	0011 1111	10	xxxx
255	0011 1111	11	xxxx
:	:	:	:
508	0111 1111	00	xxxx
509	0111 1111	01	xxxx
510	0111 1111	10	xxxx
511	0111 1111	11	xxxx

"xxxx": don't care

Note: There are 256 IAP erase pages in the HT67F60A device while there are 512 IAP erase pages in the HT67F70A device.



Read Flash Memory Procedure



Write Flash Memory Procedure

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

Data Memory

The Data Memory is an 8-bit wide RAM internal memory and is the location where temporary information is stored.

Divided into three types, the first of Data Memory 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. The third area is reserved for the LCD 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.

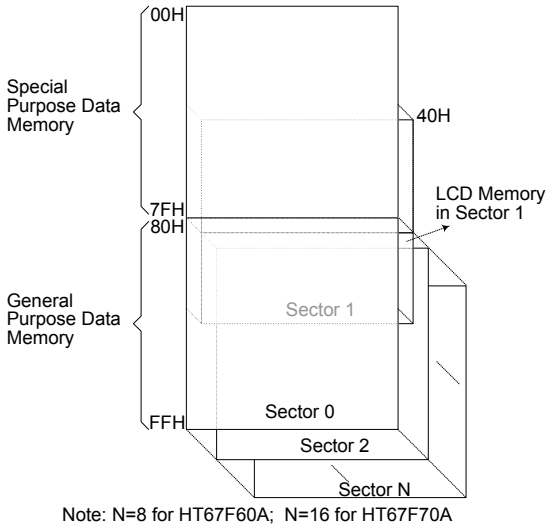
Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide Memory. Each of the Data Memory sectors is categorized into two types, the Special Purpose Data Memory and the General Purpose Data Memory.

The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the address range of the General Purpose Data Memory is from 80H to FFH.

Device	Special Puroise Data Memory		LCD Data Memory		General Puroise Data Memory	
	Capacity	Sectors	Capacity	Sectors	Capacity	Sectors
HT67F60A	192 × 8	0, 2~8: 00H~7FH 1: 40H~7FH	56 × 4	1: 80H~B7H	1024 × 8	0: 80H~FFH 2: 80H~FFH : 8: 80H~FFH
HT67F70A	192 × 8	0, 2~16: 00H~7FH 1: 40H~7FH	56 × 4	1: 80H~B7H	2048 × 8	0: 80H~FFH 2: 80H~FFH : 16: 80H~FFH

Data Memory Summary



Note: N=8 for HT67F60A; N=16 for HT67F70A

Data Memory Structure

Data Memory Addressing

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

Direct Addressing can be used in all sectors using the corresponding instruction 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 can be at least 12-bit, 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_0~8	Sector 0, 2~8	Sector 1
00H	IAR0		EEC
01H	MP0	EEA	
02H	IAR1	EED	
03H	MP1L		FC0
04H	MP1H		FC1
05H	ACC		FC2
06H	PCL	CP0C	
07H	TBLP	CP1C	
08H	TBLH	ETMC0	IFS0
09H	TBHP	ETMC1	IFS1
0AH	STATUS	ETMC2	IFS2
0BH	BP	ETMDL	IFS3
0CH	IAR2	ETMDH	IFS4
0DH	MP2L	ETMAL	
0EH	MP2H	ETMAH	
0FH		ETMBL	
10H	PAWU	ETMBH	STM1C0
11H	PAPU	STM0C0	STM1C1
12H	PA	STM0C1	STM1DL
13H	PAC	STM0DL	STM1DH
14H		STM0DH	STM1AL
15H	PBPU	STM0AL	STM1AH
16H	PB	STM0AH	STM1RP
17H	PBC	STM0RP	STM2C0
18H		CTM1C0	STM2C1
19H	PCPU	CTM1C1	STM2DL
1AH	PC	CTM1DL	STM2DH
1BH	PCC	CTM1DH	STM2AL
1CH		CTM1AL	STM2AH
1DH	PDPU	CTM1AH	STM2RP
1EH	PD	CTM0C0	
1FH	PDC	CTM0C1	
20H		CTM0DL	PAS0
21H	PEPU	CTM0DH	PAS1
22H	PE	CTM0AL	PBS0
23H	PEC	CTM0AH	PBS1
24H		PSC0	PCS0
25H	PFPU	TB0C	PCS1
26H	PF	TB1C	PDS0
27H	PFC	PSC1	PDS1
28H		SADC0	PES0
29H		SADC1	PES1
2AH	LCDC0	SADOL	PFS0
2BH	LCDC1	SADOH	PFS1
2CH		SIMC0	
2DH		SIMC1	
2EH		SIMD	
2FH	RSTC	SIMA/SIMC2	
30H	INTC0	I2CTOC	
31H	INTC1	SPIAC0	
32H	INTC2	SPIAC1	
33H	INTC3	SPIAD	
34H	MFI0	FARL	
35H	MFI1	FARH	
36H	MFI2	FD0L	
37H	MFI3	FD0H	
38H	MFI4	FD1L	
39H	INTEG	FD1H	
3AH	SMOD	FD2L	
3BH	SMOD1	FD2H	
3CH	LVRC	FD3L	
3DH	LVDC	FD3H	
3EH	WDTC	TBC2	
3FH	SMOD2		

: Unused, read as 00H

HT67F60A Special Purpose Data Memory

	Sector_0~16	Sector 0, 2~16	Sector 1
00H	IAR0		EEC
01H	MP0	EEA	
02H	IAR1	EED	
03H	MP1L		FC0
04H	MP1H		FC1
05H	ACC		FC2
06H	PCL	CP0C	
07H	TBLP	CP1C	
08H	TBLH	ETMC0	IFS0
09H	TBHP	ETMC1	IFS1
0AH	STATUS	ETMC2	IFS2
0BH	BP	ETMDL	IFS3
0CH	IAR2	ETMDH	IFS4
0DH	MP2L	ETMAL	
0EH	MP2H	ETMAH	
0FH		ETMBL	
10H	PAWU	ETMBH	STM1C0
11H	PAPU	STM0C0	STM1C1
12H	PA	STM0C1	STM1DL
13H	PAC	STM0DL	STM1DH
14H		STM0DH	STM1AL
15H	PBPU	STM0AL	STM1AH
16H	PB	STM0AH	STM1RP
17H	PBC	STM0RP	STM2C0
18H		CTM1C0	STM2C1
19H	PCPU	CTM1C1	STM2DL
1AH	PC	CTM1DL	STM2DH
1BH	PCC	CTM1DH	STM2AL
1CH		CTM1AL	STM2AH
1DH	PDPU	CTM1AH	STM2RP
1EH	PD	CTM0C0	
1FH	PDC	CTM0C1	
20H		CTM0DL	PAS0
21H	PEPU	CTM0DH	PAS1
22H	PE	CTM0AL	PBS0
23H	PEC	CTM0AH	PBS1
24H		PSC0	PCS0
25H	PFPU	TB0C	PCS1
26H	PF	TB1C	PDS0
27H	PFC	PSC1	PDS1
28H		SADC0	PES0
29H		SADC1	PES1
2AH	LCDC0	SADOL	PFS0
2BH	LCDC1	SADOH	PFS1
2CH		SIMC0	
2DH		SIMC1	
2EH		SIMD	
2FH	RSTC	SIMA/SIMC2	
30H	INTC0	I2CTOC	
31H	INTC1	SPIAC0	
32H	INTC2	SPIAC1	
33H	INTC3	SPIAD	
34H	MFI0	FARL	
35H	MFI1	FARH	
36H	MFI2	FD0L	
37H	MFI3	FD0H	
38H	MFI4	FD1L	
39H	INTEG	FD1H	
3AH	SMOD	FD2L	
3BH	SMOD1	FD2H	
3CH	LVRC	FD3L	
3DH	LVDC	FD3H	
3EH	WDTC	TBC2	
3FH	SMOD2		

: Unused, read as 00H

HT67F70A Special Purpose Data Memory

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, IAR2

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

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

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L and 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 data sectors according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all data sectors using the corresponding instruction which can address all available data memory space.

Indirect Addressing Program Example

Example 1

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

Example 2

```
data .section at 01F0H '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 IAR0               ; clear the data at address defined by MP1
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 RAM 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.

Bank Pointer – BP

Depending upon which device is used, the Program Memory is divided into several banks. Selecting the required Program Memory area is achieved using the Bank Pointer. The Bank Pointer 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 memiry bank pointer bits will occur.

Device	Bit							
	7	6	5	4	3	2	1	0
HT67F60A	—	—	—	—	—	—	—	BP0
HT67F70A	—	—	—	—	—	—	BP1	BP0

BP Register List

BP Register – HT67F60A

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	BP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **BP0**: Program Memory bank point bit 0
0: Bank 0
1: Bank 1

BP Register – HT67F70A

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	BP1	BP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **BP1~BP0**: Program Memory bank point bit 1~0
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 pointer and indicates the location where the table data is located. Their value must be setup before any table read commands are executed. Their value can be changed, for example using the "INC" or "DEC" instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), SC flag, CZ flag, 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.
- **SC** is the result of the "XOR" operation which is performed by the OV flag and the MSB of the current instruction operation result.
- **CZ** is the operational result of different flags for different instuctions. Refer to register definitions for more details.

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

STATUS Register

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

"x": unknown

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

Bit 6 **CZ**: The the operational result of different flags for different instuctions.

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 willl not be affected.

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

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

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 substraction

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

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

EEPROM Data Memory

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

Device	Capacity	Address
HT67F60A	128 × 8	00H ~ 7FH
HT67F70A		

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 128×8 bits. 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 data register in sector 0 and a single control register in sector 1.

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in sector 0, they can be directly accessed in the same was as any other Special Function Register. The EEC register, however, being located in sector 1, can be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pair and Indirect Addressing Register, IAR1 or IAR2. Because the EEC control register is located at address 40H in sector 1, the Memory Pointer low byte register, MP1L or MP2L, must first be set to the value 40H and the Memory Pointer high byte register, MP1H or MP2H, set to the value, 01H, before any operations on the EEC register are executed.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEA	—	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM Registers List

EEA Register

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~bit0

EED Register

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

Bit 7~0 **D7~D0**: Data EEPROM data bit 7~bit0

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: The WREN, WR, RDEN and RD can not be set to "1" at the same time in one instruction. The WR and RD can not be set to "1" at the same time.

Reading Data from the EEPROM

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

Writing Data to the EEPROM

To write data to the EEPROM, the EEPROM address of the data to be written must first be placed in the EEA register and the data placed in the EED register. Then 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 consecutively. 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.

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be Periodic by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Memory Pointer high byte register 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 Example

Reading data from the EEPROM – polling method

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

Writing Data to the EEPROM – polling method

```
MOV  A, EEPROM_ADRES            ; user defined address
MOV  EEA, A
MOV  A, EEPROM_DATA             ; user defined data
MOV  EED, A
MOV  A, 040H                    ; setup memory pointer low byte MP1L
MOV  MP1L, A                    ; MP1L points to EEC register
MOV  A, 01H                     ; setup Memory Pointer high byte MP1H
MOV  MP1H, A
CLR  EMI
SET  IAR1.3                     ; set WREN bit, enable write operations
SET  IAR1.2                     ; start Write Cycle - set WR bit
SET  EMI
BACK:
SZ   IAR1.2                     ; check for write cycle end
JMP  BACK
CLR  IAR1                       ; disable EEPROM write
CLR  MP1H
```

Oscillator

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

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. 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. All oscillator options are selected through register programming. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

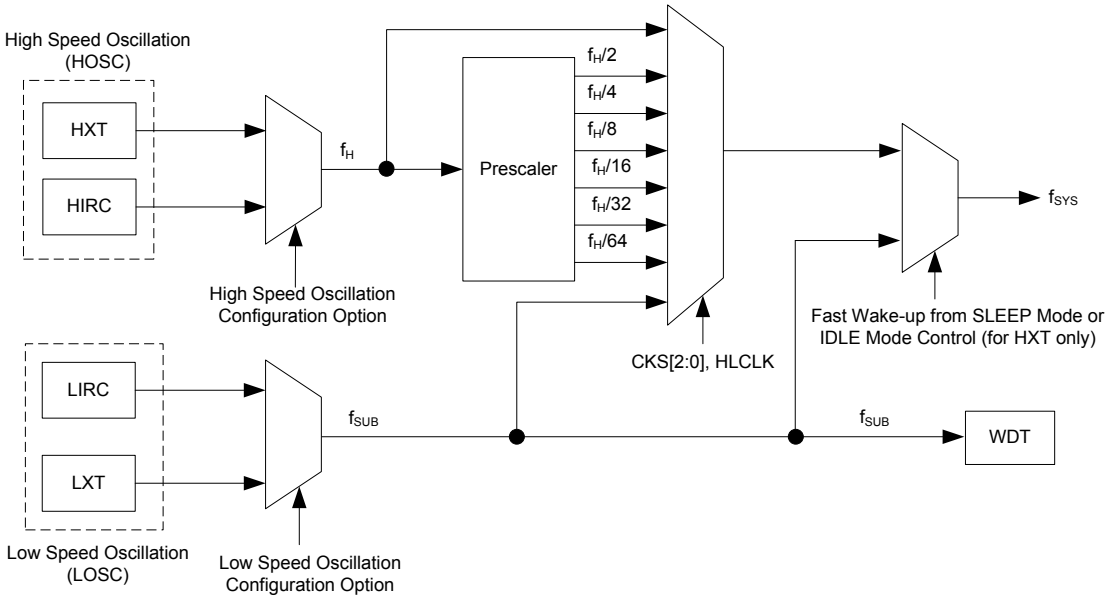
Type	Name	Freq.	Pins
External High Speed Crystal	HXT	400kHz~20 MHz	OSC1/OSC2
Internal High Speed RC	HIRC	4/8/12 MHz	—
External Low Speed Crystal	LXT	32.768 kHz	XT1/XT2
Internal Low Speed RC	LIRC	32 kHz	—

Oscillator Types

System Clock Configurations

There are four methods of generating the system clock, two high speed oscillator and two low speed oscillators. The high speed oscillators are the external crystal/ceramic and the internal 4/8/12MHz RC oscillator. The two low speed oscillators are the internal 32 kHz RC oscillator and the external 32.768 kHz crystal oscillator. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and CKS2~CKS0 bits in the SMOD register and as the system clock can be dynamically selected.

The actual source clock used for the low speed oscillators is chosen via the configuration option. The frequency of the slow speed or high speed system clock is determined using the HLCLK bit and CKS2~CKS0 bits in the SMOD 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.

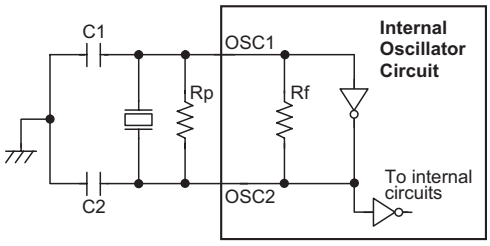


System Clock Configurations

External Crystal/Ceramic Oscillator – HXT

The External Crystal/Ceramic System Oscillator is one of the high frequency oscillator choices, which is selected via configuration option. 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. Rp 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 Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
12MHz	0 pF	0 pF
8MHz	0 pF	0 pF
4MHz	0 pF	0 pF
1MHz	100 pF	100 pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 4/8/12MHz. 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. As a result, at a power supply of 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 4MHz, 8MHz or 12MHz will have a tolerance within 2%. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins are free for use as normal I/O pins.

External 32.768 kHz Crystal Oscillator – LXT

The External 32.768 kHz Crystal System Oscillator is one of the low frequency oscillator choices, which is selected via a configuration option. This clock source has a fixed frequency of 32.768 kHz and requires a 32.768 kHz crystal to be connected between pins XT1 and XT2. The external resistor and capacitor components connected to the 32.768 kHz 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. During power-up there is a time delay associated with the LXT oscillator waitingi for it to start-up.

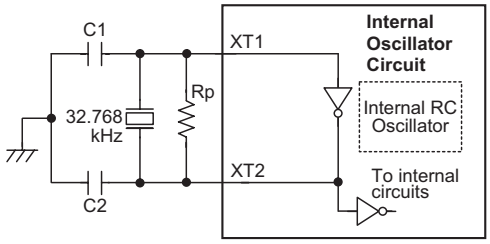
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE Mode. To do this, another clock, independent of the system clock, must be provided.

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. The external parallel feedback resistor, Rp, is required. Note that the wire connected between the 32.768 kHz crystal and the XT1/ XT2 pins should be kept as short as possible to minimise the stray noise interference.

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

- If the LXT oscillator is not used for any clock source, the XT1/XT2 pins can be used as normal I/O pins.
- If the LXT oscillator is used for any clock source, the 32.768 kHz 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. Rp, C1 and C2 are required.
2. Although not shown pins have aparasitic capacitance of around 7pF.

External LXT Oscillator

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

Crystal Recommended Capacitor Values

LXT Oscillator Low Power Function

The LXT oscillator can function in one of two modes, the Quick Start Mode and the Low-Power Mode. The mode selection is executed using the LXTLP bit in the SMOD2 register

LXTLP	LXT Operating Mode
0	Quick Start
1	Low Power

SMOD2 Register

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **LXTLP**: LXT Low Power control
0: Disable – Quick Start Mode
1: Enable – Low Power Mode

After power on the LXTLP bit will be automatically cleared to zero to ensure that the LXT oscillator is in the Quick Start operating mode. In the Quick Start 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 setting the LXTLP bit high. The oscillator will continue to run but with reduced current consumption, as the higher current consumption is only required during the LXT oscillator start-up. In power sensitive applications, such as battery applications, where power consumption must be kept to a minimum, it is therefore recommended that the application program sets the LXTLP bit high about 2 seconds after power-on.

It should be noted that, no matter what condition the LXTLP bit is set to, the LXT oscillator will 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 32 kHz System Oscillator is one of the low frequency oscillator choices, which is selected via a configuration option. It is a fully integrated RC oscillator with a typical frequency of 32 kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 5V and at a temperature of 25˚C degrees, the fixed oscillation frequency of 32 kHz will have a tolerance within 3%.

Supplementary Oscillators

The low speed oscillators, in addition to providing a system clock source are also used to provide a clock source to two other device functions. These are the Watchdog Timer and the Time Base Interrupts.

Operating Modes and System Clocks

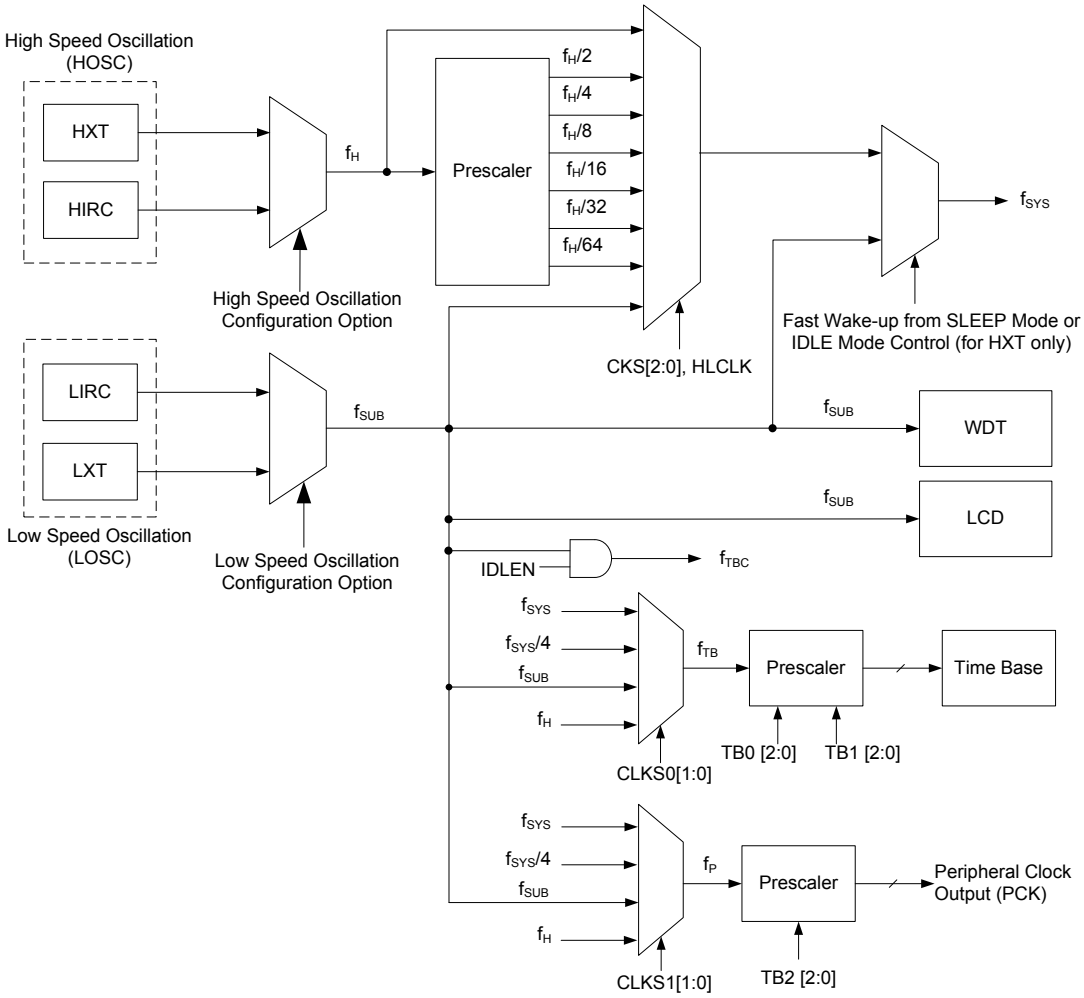
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their 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

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

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

The f_{SUB} clock is used to provide a substitute clock for the microcontroller just after a wake-up has occurred to enable faster wake-up times. The f_{SUB} clock is also used to provide the clock source for time base and watchdog timer functions.



System Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillation can be stopped to conserve the power. Therefore there is no f_H~$f_H/64$, for peripheral circuit to use.

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 NORMAL Mode and SLOW Mode. The remaining four modes, the SLEEP0, SLEEP1, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.

Operating Mode	Description			
	CPU	f_{SYS}	f_{TBC}	f_{SUB}
NORMAL	On	f_H~f_H/64	On	On
SLOW	On	f_{SUB}	On	On
IDLE0	Off	Off	On	On
IDLE1	Off	On	On	On
SLEEP0	Off	Off	Off	Off
SLEEP1	Off	Off	Off	On

NORMAL 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 and HLCLK bits in the SMOD 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 one of the low speed oscillators, either the LXT or the LIRC. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW Mode, the f_H is off.

SLEEP0 Mode

The SLEEP0 Mode is entered when an HALT instruction is executed and the IDLEN bit in the SMOD register is low. In the SLEEP0 mode the CPU will be stopped and the f_{SUB} clock will be stopped too, and the Watchdog Timer function is disabled. In this mode, the LVDEN is must set to “0”. If the LVDEN is set to “1”, it won’t enter the SLEEP0 Mode.

SLEEP1 Mode

The SLEEP1 Mode is entered when an HALT instruction is executed and the IDLEN bit in the SMOD register is low. In the SLEEP1 mode the CPU will be stopped. However the f_{SUB} will continue to operate if the LVDEN is “1” or the Watchdog Timer function is enabled.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the SMOD1 register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU but some peripheral functions will remain operational such as the Watchdog Timer and TMs. In the IDLE0 Mode, the system oscillator will be stopped while the f_{SUB} clock will be on.

IDLE1 Mode

The IDLE1 Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the SMOD1 register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU but may continue to provide a clock source to keep some peripheral functions operational such as the Watchdog Timer and TMs. In the IDLE1 Mode, the system oscillator will continue to run, and this system oscillator may be high speed or low speed system oscillator. In the IDLE1 Mode the f_{SUB} clock will also be on.

Control Registers

The registers, SMOD and SMOD1, are used to control the system clock and the corresponding oscillator configurations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SMOD	CKS2	CKS1	CKS0	FSTEN	LTO	HTO	IDLEN	HLCLK
SMOD1	FSYSON	—	—	—	—	LVRF	LRF	WRF

System Operating Mode Control Registers List

SMOD Register

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	FSTEN	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	0	0	0	0	0	0	1	1

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

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source, which can be either the LXT or the LIRC, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 **FSTEN**: Fast Wak-up control (only for HXT)
0: Disable
1: Enable

This is the Fast Wake-up Control bit which determines if the f_{SUB} clock source is initially used after the device wakes up. When the bit is high, the f_{SUB} clock source can be used as a temporary system clock to provide a faster wake up time as the f_{SUB} clock is available.

Bit 3 **LTO**: Low speed system oscillator ready flag
0: Not ready
1: Ready

This is the low speed sysem oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP0 mode but after a wake-up has occurred, the flag will change to a high level after 1024 clock cycles if the LXT oscillator is used or 1~2 clock cycles if the LIRC oscillator is used.

Bit 2 **HTO**: High speed system oscillator ready flag
0: Not ready
1: Ready

This is the high speed sysem oscillator ready flag which indicates when the high speed system oscillator is stable. The flag is cleared to “0” by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable. Therefore this flag will always be read as “1” by application program after device power-on. The flag will be low when in the SLEEP or IDLE0 mode but after a wake-up has occurred, the flag will change to a high level after 1024 clock cycles if the HXT oscillator is used or 15~16 clock cycles if the HIRC oscillator is used.

Bit 1 **IDLEN**: IDLE Mode control
0: Disable
1: Enable
This is the IDLE mode control bit and determines what happens when the HALT instruction is executed. If this bit is high, when a HALT instruction is executed, the device will enter the IDLE mode. In the IDLE1 mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational if the FSYSON bit is high. If the FSYSON bit is low, the CPU and the system clock will stop in IDLE0 mode. If the bit is low, the device will enter the SLEEP mode when a HALT instruction is executed.

Bit 0 **HLCLK**: System clock selection
0: $f_H/2$~$f_H/64$ or f_{SUB}
1: f_H
This bit is used to select if the f_H clock or the $f_H/2$~$f_H/64$ or f_{SUB} clock is used as the system clock. When the bit is high, the f_H clock will be selected and if low the $f_H/2$~$f_H/64$ or f_{SUB} clock will be selected. When the system clock switches from the f_H clock to the f_{SUB} clock, the f_H clock will be automatically switched off to conserve power.

SMOD1 Register

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

"x": unknown

Bit 7 **FSYSON**: System clock f_{SYS} control in IDLE mode
0: Disable
1: Enable

Bit 6~4 Unimplemented, read as "0"

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

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

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

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

Fast Wake-up

To minimise power consumption the device can enter the SLEEP or IDLE0 Mode, where the system clock source to the device will be stopped. However when the device is woken up again, it can take a considerable time for the original system oscillator to restart, stabilize and allow normal operation to resume. To ensure the device is up and running as fast as possible a Fast Wake-up function is provided, which allows f_{SUB}, namely either the LXT or LIRC oscillator, to act as a temporary clock to first drive the system until the original system oscillator has stabilised. As the clock source for the Fast Wake-up function is f_{SUB}, the Fast Wake-up function is only available in the SLEEP1 and IDLE0 modes. When the device is woken up from the SLEEP0 mode, the FastWake-up function has no effect because the f_{SUB} clock is stopped. The FastWake-up enable/disable function is controlled using the FSTEN bit in the SMOD1 register.

If the HXT oscillator is selected as the NORMAL Mode system clock and if the Fast Wake-up function is enabled, then it will take one to two t_{SUB} clock cycles of the LXT or LIRC oscillator for the system to wake-up. The system will then initially run under the f_{SUB} clock source until 1024 HXT clock cycles have elapsed, at which point the HTO flag will switch high and the system will switch over to operating from the HXT oscillator.

If the HIRC or LIRC oscillator is used as the system oscillator, then it will take 15~16 clock cycles of the HIRC oscillator or 1~2 clock cycles of the LIRC osrillator respectively to wake up the system from the SLEEP or IDLE0 Mode. The Fast Wake-up bit, FSTEN will have no effect in these cases.

System Oscillator	FSTEN Bit	Wake-up Time (SLEEP0 Mode)	Wake-up Time (SLEEP1 Mode)	Wake-up Time (IDLE0 Mode)	Wake-up Time (IDLE1 Mode)
HXT	0	1024 HXT cycles	1024 HXT cycles		1~2 HXT cycles
	1	1024 HXT cycles	1~2 f_{SUB} cycles (System runs with f_{SUB} first for 1024 HXT cycles and then switches over to run with the HXT clock)		1~2 HXT cycles
HIRC	x	15~16 HIRC cycles	15~16 HIRC cycles		1~2 HIRC cycles
LIRC	x	1~2 LIRC cycles	1~2 LIRC cycles		1~2 LIRC cycles
LXT	x	1024 LXT cycles	1024 LXT cycles		1~2 LXT cycles

Wake-up Times

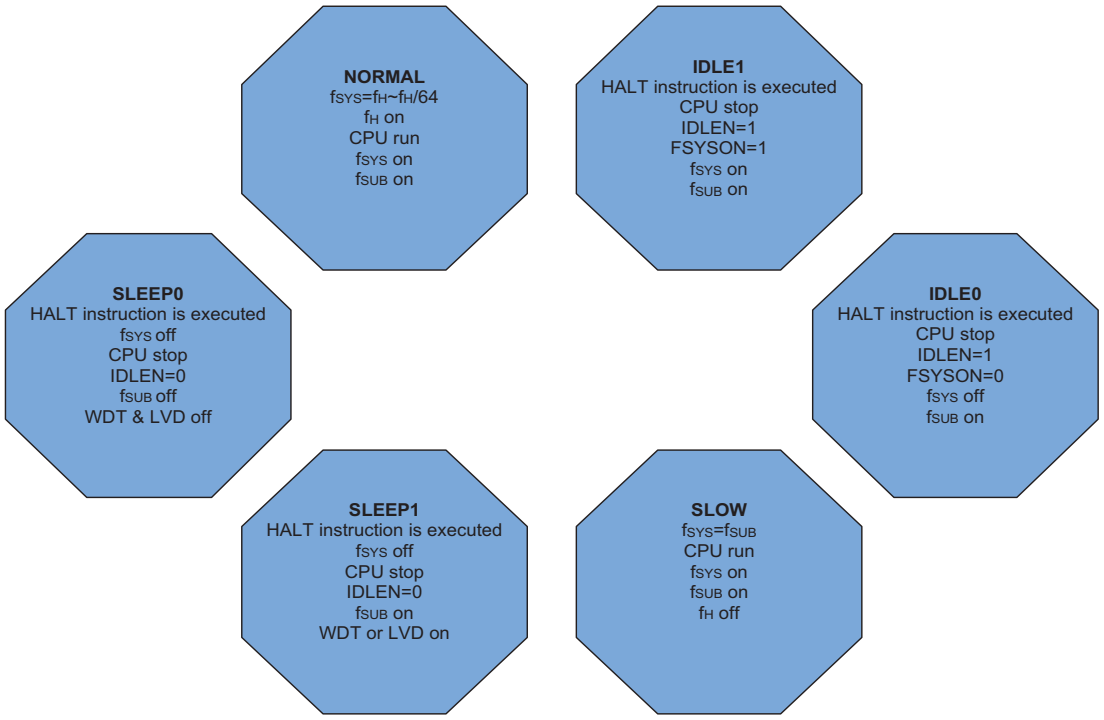
Note that if the Watchdog Timer is disabled, which means that the f_{SUB} clock derived from the LXT or LIRC oscillator is off, then there will be no Fast Wake-up function available when the device wakes up from the SLEEP0 Mode.

Operating Mode Switching

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

In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/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 IDLEN bit in the SMOD register and FSYSON in the SMOD1 register.

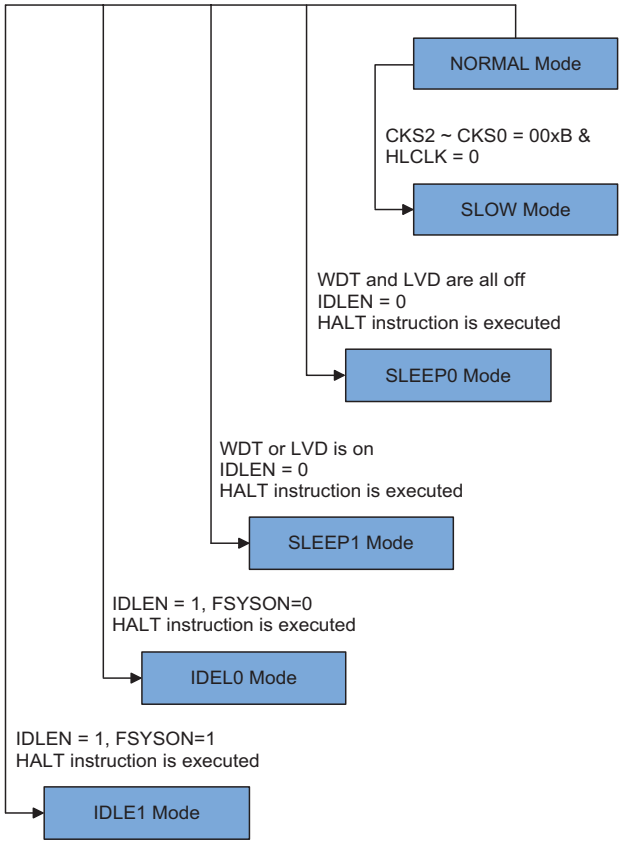
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~ $f_H/64$ or f_{SUB}. If the clock is from the f_{SUB}, the high speed clock source will stop running to conserve power. When this happens it must be noted that the $f_H/16$ and $f_H/64$ internal clock sources will also stop running, which may affect the operation of other internal functions such as the TMs. The accompanying flowchart shows what happens when the device moves between the various operating modes.



NORMAL Mode to SLOW Mode Switching

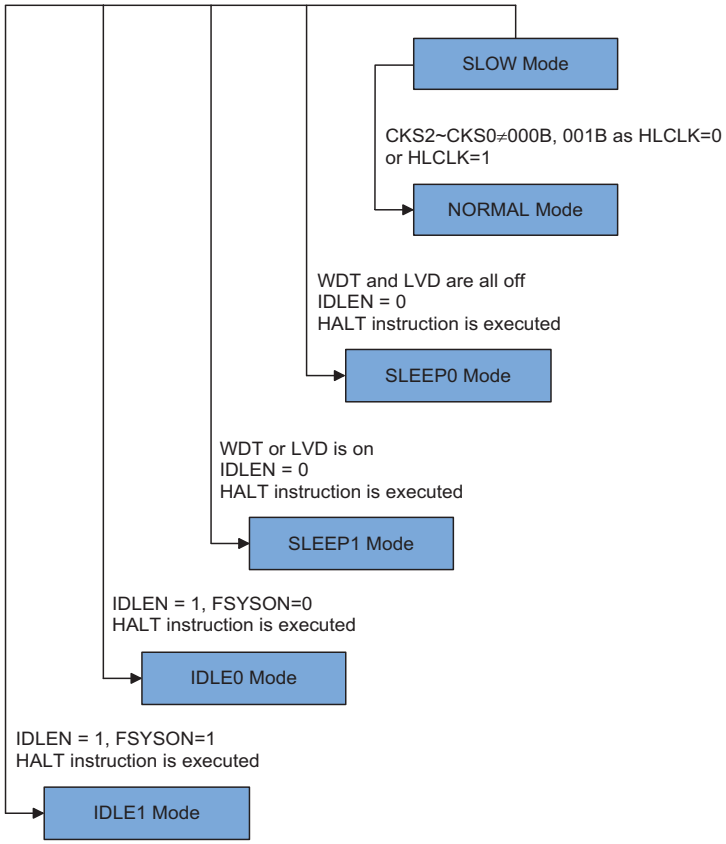
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to "0" and set the CKS2~CKS0 bits to "000" or "001" in the SMOD register. This will then use the low speed system oscillator 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 the LIRC oscillators and therefore requires these oscillators to be stable before full mode switching occurs. This is monitored using the LTO bit in the SMOD register.



SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system uses either the LXT or LIRC low speed system oscillator. To switch back to the NORMAL Mode, where the high speed system oscillator is used, the HLCLK bit should be set to "1" or HLCLK bit is "0", but CKS2~CKS0 is set to "010", "011", "100", "101", "110" or "111". As a certain amount of time will be required for the high frequency clock to stabilise, the status of the HTO bit is checked. The amount of time required for high speed system oscillator stabilization depends upon which high speed system oscillator type is used.



Entering the SLEEP0 Mode

There is only one way for the device to enter the SLEEP0 Mode and that is to execute the “HALT” instruction in the application program with the IDLEN bit in the SMOD register equal to “0” and the WDT and LVD both off. When this instruction is executed under the conditions described above, the following will occur:

- The system clock and the f_{SUB} clock will be stopped and the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and stopped as the WDT function is disabled.

Entering the SLEEP1 Mode

There is only one way for the device to enter the SLEEP1 Mode and that is to execute the “HALT” instruction in the application program with the IDLEN bit in the SMOD register equal to “0” and the WDT or LVD on. 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 but the WDT or LVD will remain with the clock source coming from the f_{SUB} clock
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting as the WDT function is enabled and the clock source is derived from the f_{SUB} clock.

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 IDLEN bit in SMOD register equal to “1” and the FSYSON bit in SMOD1 register equal to “0”. 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, but the f_{SUB} clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting as the WDT clock source is derived from the f_{SUB} clock.

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 the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in SMOD1 register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The system clock and the f_{SUB} clock will be on and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting as the WDT clock source is derived from the f_{SUB} clock.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of these devices to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 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 these devices. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to devices which have 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 configuration options have enabled the LXT or LIRC oscillator.

In the IDLE1 Mode the system oscillator is on, if the system oscillator is from the high speed system oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

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

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

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

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of these devices to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 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 these devices. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to devices which have 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 configuration options have enabled the LXT or LIRC oscillator.

In the IDLE1 Mode the system oscillator is on, if the system oscillator is from the high speed system oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

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 f_{SUB} clock. The f_{SUB} clock can be source from either the LXT or LIRC oscillator selected by a configuration option. The LIRC oscillator has an approximate frequency of 32 kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The LXT oscillator is supplied by an external 32.768 kHz crystal. 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 timeout period as well as the enable/disable operation. This register controls the overall operation of the Watchdog Timer.

WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT function enable control
10101: Disabled
01010: Enabled
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 2~3 LIRC clock cycles and the WRF bit in the SMOD1 register will be set to 1.

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

SMOD1 Register

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

"x": unknown

Bit 7 **FSYSON**: System clock f_{SYS} control in IDLE mode
Described elsewhere

Bit 6~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Described elsewhere

Bit 2 **LVRF**: LVR function reset flag
Described elsewhere

Bit 1 **LRF**: LVR control register software reset flag
Described elsewhere

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

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, 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 2~3 f_{LIRC} clock cycles. After power on these bits will have a value of 01010B.

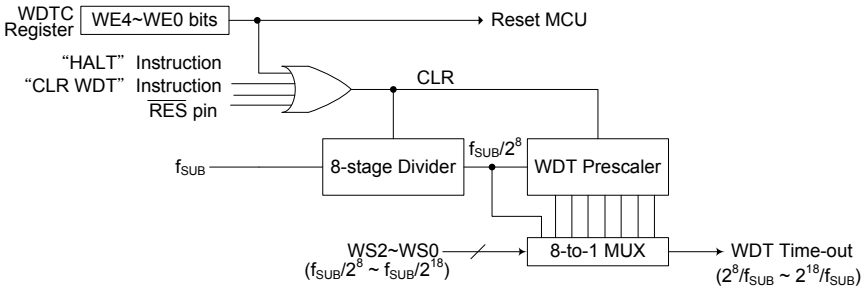
WE4 ~ WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

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. Four methods can be adopted to clear the contents of the Watchdog Timer. The first is an external hardware reset, which means a low level on the $\overline{RES}$ pin. The second is a WDT software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 field, the third is using the Watchdog Timer software clear instruction and the forth 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 contents.

The maximum time out period is when the 2^{18} division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 second for the 2^{18} division ratio, and a minimum timeout of 7.8ms 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, situations may arise where it is necessary to forcefully apply a reset condition when the device is running. One example of this is where after power has been applied and the device is already running, the $\overline{\text{RES}}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the registers remain unchanged allowing the device to proceed with normal operation after the reset line is allowed to return high.

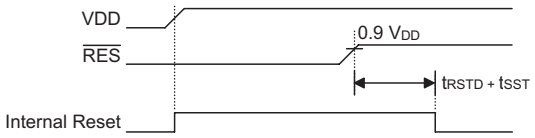
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset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

Reset Functions

There are five ways in which a microcontroller reset can occur, through events occurring both internally and externally.

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

$\overline{\text{RES}}$ Pin Reset

As the reset pin is shared with an I/O pin, the reset function must be selected using the RSTC control register. The RSTC register is used to determine that the pin is used as an external reset pin or an I/O pin. However, if the content of the RSTC register is set to any value other than 01010101B or 10101010B, it will reset the device after 2~3 f_{LIRC} clock cycles. After power on the register will have a value of 01010101B.

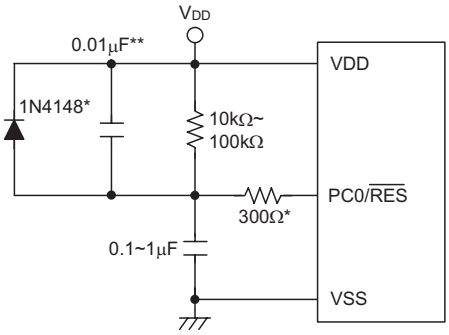
RSTC7 ~ RSTC0 Bits	Pin Function
01010101B	I/O pin
10101010B	$\overline{\text{RES}}$ pin
Any other value	Reset MCU

Internal Reset Function Control

Although the microcontroller has an internal RC reset function, if the V_{DD} power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{\text{RES}}$ pin, whose additional time delay will ensure that the $\overline{\text{RES}}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{\text{RES}}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.

For most applications a resistor connected between VDD and the $\overline{\text{RES}}$ pin and a capacitor connected between VSS and the $\overline{\text{RES}}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{\text{RES}}$ pin should be kept as short as possible to minimise any stray noise interference.

For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



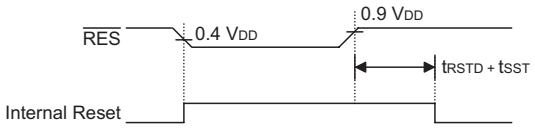
Note: * It is recommended that this component is added for added ESD protection.

** It is recommended that this component is added in environments where power line noise is significant.

External $\overline{\text{RES}}$ Circuit

More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

Pulling the $\overline{\text{RES}}$ Pin low using external hardware will also execute a device reset. In this case, as in the case of other resets, the Program Counter will reset to zero and program execution initiated from this point.



$\overline{\text{RES}}$ Reset Timing Chart

- **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 pin function control
01010101: I/O pin
10101010: External reset pin
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 2~3 LIRC clock cycles and the RSTF bit in the SMOD1 register will be set to 1.

- **SMOD1 Register**

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

"x": unknown

Bit 7 **FSYSON**: System clock f_{SYS} control in IDLE mode
Described elsewhere

Bit 6~4 Unimplemented, read as "0"

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

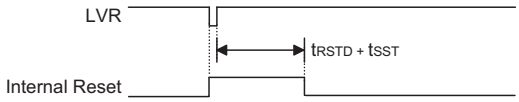
Bit 2 **LVRF**: LVR function reset flag
Described elsewhere

Bit 1 **LRF**: LVR control register software reset flag
Described elsewhere

Bit 0 **WRF**: WDT control register software reset flag
Described elsewhere

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage, V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the SMOD1 register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVD/LVR characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits have any other value, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the device after 2~3 f_{LIRC} clock cycles. When this happens, the LRF bit in the SMOD1 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 power down 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
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Other values: Generates a 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 value above, an MCU reset will generated. The reset operation will be activated after 2~3 f_{LIRC} clock cycles. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than the four defined register values above, will also result in the generation of an MCU reset. The reset operation will be activated after 2~3 f_{LIRC} clock cycles. However in this situation the register contents will be reset to the POR value.

- **SMOD1 Register**

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

"x": unknown

Bit 7 **FSYSON**: System clock f_{SYS} control in IDLE mode
Described elsewhere

Bit 6~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Described elsewhere

Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred

This bit is set to 1 when a specific low voltage reset condition occurs. Note that this bit can only be cleared to 0 by the application program.

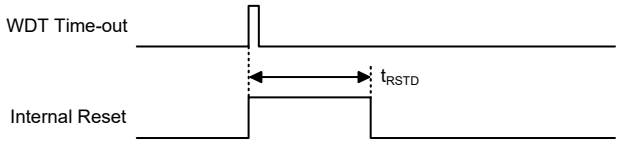
Bit 1 **LRF**: LVR control register software reset flag
0: Not occurred
1: Occurred

This bit is set to 1 by the LVRC control register contains any undefined LVR voltage register values. This in effect acts like a software-reset function. Note that this bit can only be cleared to 0 by the application program.

Bit 0 **WRF**: WDT control register software reset flag
Described elsewhere

Watchdog Time-out Reset during Normal Operation

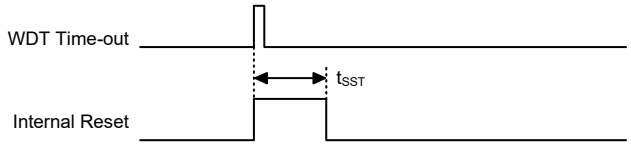
The Watchdog time-out Reset during normal operation is the same as the hardware $\overline{RES}$ pin reset except that the Watchdog time-out flag TO will be set to "1".



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 A.C. Characteristics for t_{SST} details.



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

Reset Initial Conditions

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

TO	PDF	Reset Function
0	0	Power-on reset
u	u	$\overline{RES}$ or LVR reset during NORMAL or SLOW Mode operation
1	u	WDT time-out reset during NORMAL or SLOW Mode operation
1	1	WDT time-out reset during 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	Reset Function
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Base	Clear after reset, WDT begins counting
Timer Modules	Timer Modules will be turned off
Input/Output Ports	I/O ports will be setup as inputs
Stack pointer	Stack pointer will point to the top of the stack

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

Register	HT67F60A	HT67F70A	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE or SLEEP)
IAR0	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	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	●		--xx xxxx	---- uuuu	---- uuuu	---- uuuu
TBHP		●	-xxx xxxx	---- uuuu	---- uuuu	---- uuuu
STATUS	●	●	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
BP	●		---- ---0	---- ---0	---- ---0	---- ---u
BP		●	---- --00	---- --00	---- --00	---- --uu
IAR2	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP2L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PA	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
PB	●	●	-000 1111	-000 1111	-000 1111	-uuu uuuu
PBC	●	●	-111 1111	-111 1111	-111 1111	-uuu uuuu
PCPU	●	●	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
PDPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PE	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PF	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
LCDC0	●	●	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
LCDC1	●	●	000- 0000	000- 0000	000- 0000	uuu- uuuu
RSTC	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
INTC0	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC3	●	●	-000 -000	-000 -000	-000 -000	-uuu -uuu
MFI0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI1	●	●	-000 -000	-000 -000	-000 -000	-uuu -uuu

Register	HT67F60A	HT67F70A	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE or SLEEP)
MFI2	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI3	•	•	-000 -000	-000 -000	-000 -000	-uuu -uuu
MFI4	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTEG	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
SMOD	•	•	0000 0011	0000 0011	0000 0011	uuuu uuuu
SMOD1	•	•	0--- 0x00	0--- 0x00	0--- 0x00	u--- uuuu
LVRC	•	•	0101 0101	0101 0101	0101 0101	uuuu uuuu
LVDC	•	•	--00 0000	--00 0000	--00 0000	--uu uuuu
WDTC	•	•	0101 0011	0101 0011	0101 0011	uuuu uuuu
SMOD2	•	•	---- ---0	---- ---0	---- ---0	---- ---u
EEA	•	•	-000 0000	-000 0000	-000 0000	-uuu uuuu
EED	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CP0C	•	•	-000 ---1	-000 ---1	-000 ---1	-uuu ---u
CP1C	•	•	-000 ---1	-000 ---1	-000 ---1	-uuu ---u
ETMC0	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMC1	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMC2	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMDL	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMDH	•	•	---- --00	---- --00	---- --00	---- --uu
ETMAL	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMAH	•	•	---- --00	---- --00	---- --00	---- --uu
ETMBL	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMBH	•	•	---- --00	---- --00	---- --00	---- --uu
STM0C0	•	•	0000 0---	0000 0---	0000 0---	uuuu u---
STM0C1	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0DL	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0DH	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0AL	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0AH	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0RP	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1C0	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1C1	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DL	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DH	•	•	---- --00	---- --00	---- --00	---- --uu
CTM1AL	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1AH	•	•	---- --00	---- --00	---- --00	---- --uu
CTM0C0	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0C1	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DL	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DH	•	•	---- --00	---- --00	---- --00	---- --uu
CTM0AL	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0AH	•	•	---- --00	---- --00	---- --00	---- --uu
PSC0	•	•	---- --00	---- --00	---- --00	---- --uu
TBC0	•	•	0--- -000	0--- -000	0--- -000	u--- -uuu
TBC1	•	•	0--- -000	0--- -000	0--- -000	u--- -uuu

Register	HT67F60A	HT67F70A	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE or SLEEP)
PSC1	●	●	---- --00	---- --00	---- --00	---- --uu
SADC0	●	●	0110 0000	0110 0000	0110 0000	uuuu uuuu
SADC1	●	●	-000 -000	-000 -000	-000 -000	-uuu –uuu
SADOL (ADRFS=0)	●	●	xxxx ----	xxxx ----	xxxx ----	uuuu ----
SADOL (ADRFS=1)	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
SADOH (ADRFS=0)	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
SADOH (ADRFS=1)	●	●	---- xxxx	---- uuuu	---- uuuu	---- uuuu
SIMC0	●	●	111- 000-	111- 000-	111- 000-	uuu- uuu-
SIMC1	●	●	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMC1/SIMA	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
I2CTOC	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SPIAC0	●	●	111- --0-	111- --0-	111- --0-	uuu- --u-
SPIAC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SPIAD	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	●		--00 0000	--00 0000	--00 0000	--uu uuuu
FARH		●	-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
TBC2	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
EEC	●	●	---- 0000	---- 0000	---- 0000	---- uuuu
FC0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC2	●	●	---- ---0	---- ---0	---- ---0	---- ---u
IFS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS3	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS4	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1C0	●	●	0000 0---	0000 0---	0000 0---	uuuu u---
STM1C1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1AL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1AH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1RP	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2C0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2C1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	HT67F60A	HT67F70A	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE or SLEEP)
STM2DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2AL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2AH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2RP	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
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
PDS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS1	●	●	0000 0000	0000 0000	0000 0000	uuuu 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	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu

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

Input/Output Ports

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

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
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PBPU	—	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PB	—	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	—	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0
PF	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0

I/O Registers List

"—": Unimplemented, read as "0"

PAWUn: PA wake-up function control
0: Disable
1: Enable

PAPUn/PBPUn/PCPUn/PDPUn/PEPUn/PFPUn: I/O Pin Pull-high function control
0: Disable
1: Enable

PAn/PBn/PCn/PDn/PEn/PFn: I/O Pin Data bit
0: Data 0
1: Data 1

PACn/PBCn/PCCn/PDCn/PECn/PFCn: I/O Pin type selection
0: Output
1: Input

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.

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.

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 I/O Port A bit 7 ~ bit 0 wake-up function Control
0: Disable
1: Enable

I/O Port Control Registers

Each Port has its own control register, known as PAC~PFC, which controls the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. 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.

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 i, 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 I²C 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, as well as the Timer Module input.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. 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. To correctly deselect the pn-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PA3S1	PA3S0	PA2S1	PA2S0	PA1S1	PA1S0	PA0S1	PA0S0
PAS1	PA7S1	PA7S0	PA6S1	PA6S0	PA5S1	PA5S0	PA4S1	PA4S0
PBS0	PB3S1	PB3S0	PB2S1	PB2S0	PB1S1	PB1S0	PB0S1	PB0S0
PBS1	—	—	PB6S1	PB6S0	PB5S1	PB5S0	PB4S1	PB4S0
PCS0	PC3S1	PC3S0	PC2S1	PC2S0	PC1S1	PC1S0	PC0S1	PC0S0
PCS1	PC7S1	PC7S0	PC6S1	PC6S0	PC5S1	PC5S0	PC4S1	PC4S0
PDS0	PD3S1	PD3S0	PD2S1	PD2S0	PD1S1	PD1S0	PD0S1	PD0S0
PDS1	PD7S1	PD7S0	PD6S1	PD6S0	PD5S1	PD5S0	PD4S1	PD4S0
PES0	PE3S1	PE3S0	PE2S1	PE2S0	PE1S1	PE1S0	PE0S1	PE0S0
PES1	PE7S1	PE7S0	PE6S1	PE6S0	PE5S1	PE5S0	PE4S1	PE4S0
PFS0	PF3S1	PF3S0	PF2S1	PF2S0	PF1S1	PF1S0	PF0S1	PF0S0
PFS1	PF7S1	PF7S0	PF6S1	PF6S0	PF5S1	PF5S0	PF4S1	PF4S0
IFS0	PINTBS	—	—	—	INT3S	INT2S	INT1S	INT0S
IFS1	—	STCK0S	STP0IS	ETCKS	ETPIBS	ETPIAS	CTCK0S	—
IFS2	—	—	STCK2S	STP2IS	STCK1S	STP1IS	CTCK1S	—
IFS3	—	—	SDIAS	SDIS	SCKAS	SCKS	SCSAS	SCSS
IFS4	—	—	—	C1NS	C1PS	—	C0NS	C0PS

Pin-shared Function Selection Registers List

PAS0 Register

Bit	7	6	5	4	3	2	1	0
Name	PA3S1	PA3S0	PA2S1	PA2S0	PA1S1	PA1S0	PA0S1	PA0S0
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 **PA3S1~PA3S0**: PA3 pin function selection
00: PA3/CTCK0
01: SDI/SDA
10: VREF
11: AN1

Bit 5~4 **PA2S1~PA2S0**: PA2 pin function selection
00/01/10: PA2
11: AN4

Bit 3~2 **PA1S1~PA1S0**: PA1 pin function selection
00: PA1
01: SDO
10: CTP0
11: AN0

Bit 1~0 **PA0S1~PA0S0**: PA0 pin function selection
00/01/10: PA0
11: AN5

PAS1 Register

Bit	7	6	5	4	3	2	1	0
Name	PA7S1	PA7S0	PA6S1	PA6S0	PA5S1	PA5S0	PA4S1	PA4S0
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 **PA7S1~PA7S0**: PA7 pin function selection
00/10: PA7/INT1/STP0I
01: STP0
11: AN7

Bit 5~4 **PA6S1~PA6S0**: PA6 pin function selection
00/01/10: PA6/INT0/ETCK
11: AN6

Bit 3~2 **PA5S1~PA5S0**: PA5 pin function selection
00: PA5/ETPIB
01: $\overline{SCS}$
10: ETPB
11: AN3

Bit 1~0 **PA4S1~PA4S0**: PA4 pin function selection
00: PA4/ETPIA/$\overline{PINT}$
01: SCK/SCL
10: ETPA
11: AN2

PBS0 Register

Bit	7	6	5	4	3	2	1	0
Name	PB3S1	PB3S0	PB2S1	PB2S0	PB1S1	PB1S0	PB0S1	PB0S0
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 **PB3S1~PB3S0**: PB3 pin function selection
00/01: PB3/INT3
10: CTP1
11: AN11

Bit 5~4 **PB2S1~PB2S0**: PB2 pin function selection
00/01/10: PB2/INT2/STCK0
11: AN10

Bit 3~2 **PB1S1~PB1S0**: PB1 pin function selection
00/10: PB1
01: SCKA
11: AN9

Bit 1~0 **PB0S1~PB0S0**: PB0 pin function selection
00/10: PB0/CTCK1
01: $\overline{\text{SCSA}}$
11: AN8

PBS1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PB6S1	PB6S0	PB5S1	PB5S0	PB4S1	PB4S0
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 **PB6S1~PB6S0**: PB6 pin function selection
00/01/10: PB6
11: C2

Bit 3~2 **PB5S1~PB5S0**: PB5 pin function selection
00/01/10: PB5
11: C1

Bit 1~0 **PB4S1~PB4S0**: PB4 pin function selection
00/01/10: PB4
11: V2

PCS0 Register

Bit	7	6	5	4	3	2	1	0
Name	PC3S1	PC3S0	PC2S1	PC2S0	PC1S1	PC1S0	PC0S1	PC0S0
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 **PC3S1~PC3S0**: PC3 pin function selection
00: PC3/STCK1
01: PCK
10: C0P
11: SEG3

Bit 5~4 **PC2S1~PC2S0**: PC2 pin function selection
00: PC2/STP2I
01: STP2
10: C1X
11: SEG2

Bit 3~2 **PC1S1~PC1S0**: PC1 pin function selection
00/01: PC1/STCK2/$\overline{\text{PINT}}$
10: C1N
11: SEG1

Bit 1~0 **PC0S1~PC0S0**: PC0 pin function selection
00/01: PC0
10: C1P
11: SEG0

PCS1 Register

Bit	7	6	5	4	3	2	1	0
Name	PC7S1	PC7S0	PC6S1	PC6S0	PC5S1	PC5S0	PC4S1	PC4S0
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 **PC7S1~PC7S0**: PC7 pin function selection
00: PC7/STCK0
01: $\overline{\text{SCSA}}$
10: C0N
11: SEG7

Bit 5~4 **PC6S1~PC6S0**: PC6 pin function selection
00: PC6
01: SCKA
10: CTP1
11: SEG6

Bit 3~2 **PC5S1~PC5S0**: PC5 pin function selection
00: PC5/CTCK1
01: SDIA
10: C0X
11: SEG5

Bit 1~0 **PC4S1~PC4S0**: PC4 pin function selection
00: PC4/STP1I
01: SDOA
10: STP1
11: SEG4

PDS0 Register

Bit	7	6	5	4	3	2	1	0
Name	PD3S1	PD3S0	PD2S1	PD2S0	PD1S1	PD1S0	PD0S1	PD0S0
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 **PD3S1~PD3S0**: PD3 pin function selection
00: PD3
01: SDO
10: C1N
11: SEG11

Bit 5~4 **PD2S1~PD2S0**: PD2 pin function selection
00: PD2/ETPIB
01: SDI/SDA
10: ETPB
11: SEG10

Bit 3~2 **PD1S1~PD1S0**: PD1 pin function selection
00: PD1/ETCK
01: SCK/SCL
10: C1P
11: SEG9

Bit 1~0 **PD0S1~PD0S0**: PD0 pin function selection
00: PD0/STP0I
01: $\overline{SCS}$
10: STP0
11: SEG8

PDS1 Register

Bit	7	6	5	4	3	2	1	0
Name	PD7S1	PD7S0	PD6S1	PD6S0	PD5S1	PD5S0	PD4S1	PD4S0
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 **PD7S1~PD7S0**: PD7 pin function selection
00: PD7/INT0/STP2I
01: STP2
10: C0X
11: SEG15

Bit 5~4 **PD6S1~PD6S0**: PD6 pin function selection
00/01: PD6/INT1/STCK2
10: C0N
11: SEG14

Bit 3~2 **PD5S1~PD5S0**: PD5 pin function selection
00: PD5/INT2
01: CTP0
10: C0P
11: SEG13

Bit 1~0 **PD4S1~PD4S0**: PD4 pin function selection
00: PD4/INT3/CTCK0/ETPIA
01: ETPA
10: C1X
11: SEG12

PES0 Register

Bit	7	6	5	4	3	2	1	0
Name	PE3S1	PE3S0	PE2S1	PE2S0	PE1S1	PE1S0	PE0S1	PE0S0
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 **PE3S1~PE3S0**: PE3 pin function selection
00/01/10: PE3
11: SEG19

Bit 5~4 **PE2S1~PE2S0**: PE2 pin function selection
00/01/10: PE2
11: SEG18

Bit 3~2 **PE1S1~PE1S0**: PE1 pin function selection
00: PE1/STP1I
01: SDIA
10: STP1
11: SEG17

Bit 1~0 **PE0S1~PE0S0**: PE0 pin function selection
00/10: PE0/STCK1
01: SDOA
11: SEG16

PES1 Register

Bit	7	6	5	4	3	2	1	0
Name	PE7S1	PE7S0	PE6S1	PE6S0	PE5S1	PE5S0	PE4S1	PE4S0
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 **PE7S1~PE7S0**: PE7 pin function selection
00/01/10: PE7
11: SEG23

Bit 5~4 **PE6S1~PE6S0**: PE6 pin function selection
00/01/10: PE2
11: SEG22

Bit 3~2 **PE5S1~PE5S0**: PE5 pin function selection
00/01/10: PE5
11: SEG21

Bit 1~0 **PE4S1~PE4S0**: PE4 pin function selection
00/01/10: PE4
11: SEG20

PFS0 Register

Bit	7	6	5	4	3	2	1	0
Name	PF3S1	PF3S0	PF2S1	PF2S0	PF1S1	PF1S0	PF0S1	PF0S0
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 **PF3S1~PF3S0**: PF3 pin function selection
00/01/10: PF3
11: SEG27

Bit 5~4 **PF2S1~PF2S0**: PF2 pin function selection
00/01/10: PF2
11: SEG26

Bit 3~2 **PF1S1~PF1S0**: PF1 pin function selection
00/01/10: PF1
11: SEG25

Bit 1~0 **PF0S1~PF0S0**: PF0 pin function selection
00/01/10: PF0
11: SEG24

PFS1 Register

Bit	7	6	5	4	3	2	1	0
Name	PF7S1	PF7S0	PF6S1	PF6S0	PF5S1	PF5S0	PF4S1	PF4S0
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 **PF7S1~PF7S0**: PF7 pin function selection
00/01/10: PF7
11: SEG31

Bit 5~4 **PF6S1~PF6S0**: PF6 pin function selection
00/01/10: PF2
11: SEG30

Bit 3~2 **PF5S1~PF5S0**: PF5 pin function selection
00/01/10: PF5
11: SEG29

Bit 1~0 **PF4S1~PF4S0**: PF4 pin function selection
00/01/10: PF4
11: SEG28

IFS0 Register

Bit	7	6	5	4	3	2	1	0
Name	PINTBS	—	—	—	INT3S	INT2S	INT1S	INT0S
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **PINTBS**: $\overline{\text{PINT}}$ input source pin selection
0: PC1
1: PA4

Bit 6~4 Unimplemented, read as "0"

Bit 3 **INT3S**: INT3 input source pin selection
0: PB3
1: PD4

Bit 2 **INT2S**: INT2 input source pin selection
0: PB2
1: PD5

Bit 1 **INT1S**: INT1 input source pin selection
0: PA7
1: PD6

Bit 0 **INT0S**: INT0 input source pin selection
0: PA6
1: PD7

IFS1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	STCK0S	STP0IS	ETCKS	ETPIBS	ETPIAS	CTCK0S	—
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 **STCK0S**: STCK0 input source pin selection
0: PB2
1: PC7

Bit 5 **STP0IS**: STP0I input source pin selection
0: PA7
1: PD0

Bit 4 **ETCKS**: ETCK input source pin selection
0: PA6
1: PD1

Bit 3 **ETPIBS**: ETPIB input source pin selection
0: PA5
1: PD2

Bit 2 **ETPIAS**: ETPIA input source pin selection
0: PA4
1: PD4

Bit 1 **CTCK0S**: CTCK0 input source pin selection
0: PA3
1: PD4

Bit 0 Unimplemented, read as "0"

IFS2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	STCK2S	STP2IS	STCK1S	STP1IS	CTCK1S	—
R/W	—	—	R/W	R/W	R/W	R/W	R/W	—
POR	—	—	0	0	0	0	0	—

Bit 7~6 Unimplemented, read as "0"

Bit 5 **STCK2S**: STCK2 input source pin selection
0: PC1
1: PD6

Bit 4 **STP2IS**: STP2I input source pin selection
0: PC2
1: PD7

Bit 3 **STCK1S**: STCK1 input source pin selection
0: PC3
1: PE0

Bit 2 **STP1IS**: STP1I input source pin selection
0: PC4
1: PE1

Bit 1 **CTCK1S**: CTCK1 input source pin selection
0: PB0
1: PC5

Bit 0 Unimplemented, read as "0"

IFS3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	SDIAS	SDIS	SCKAS	SCKS	SCSAS	SCSS
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 **SDIAS**: SDIA input source pin selection
0: PC5
1: PE1

Bit 4 **SDIS**: SDI input source pin selection
0: PA3
1: PD2

Bit 3 **SCKAS**: SCKA input source pin selection
0: PB1
1: PC6

Bit 2 **SCKS**: SCK/SCL input source pin selection
0: PA4
1: PD1

Bit 1 **SCSAS**: $\overline{\text{SCSA}}$ input source pin selection
0: PB0
1: PC7

Bit 0 **SCSS**: $\overline{\text{SCS}}$ input source pin selection
0: PA5
1: PD0

IFS4 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	C1NS	C1PS	—	C0NS	C0PS
R/W	—	—	—	R/W	R/W	—	R/W	R/W
POR	—	—	—	0	0	—	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4 **C1NS**: C1N input source pin selection
0: PC1
1: PD3

Bit 3 **C1PS**: C1P input source pin selection
0: PC0
1: PD1

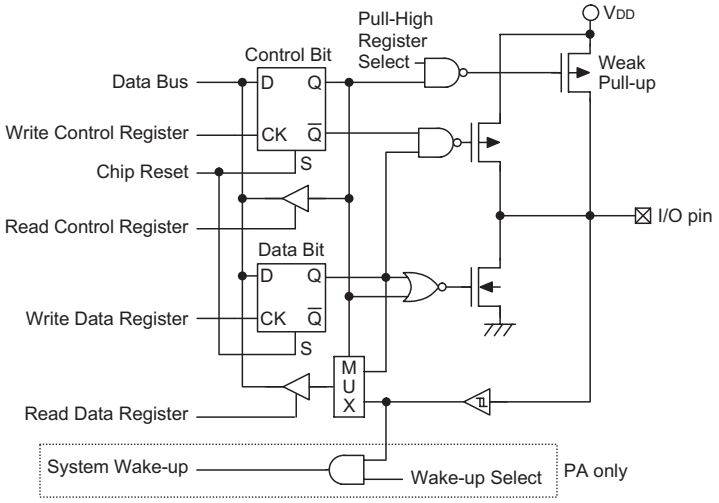
Bit 2 Unimplemented, read as "0"

Bit 1 **C0NS**: C0N input source pin selection
0: PC7
1: PD6

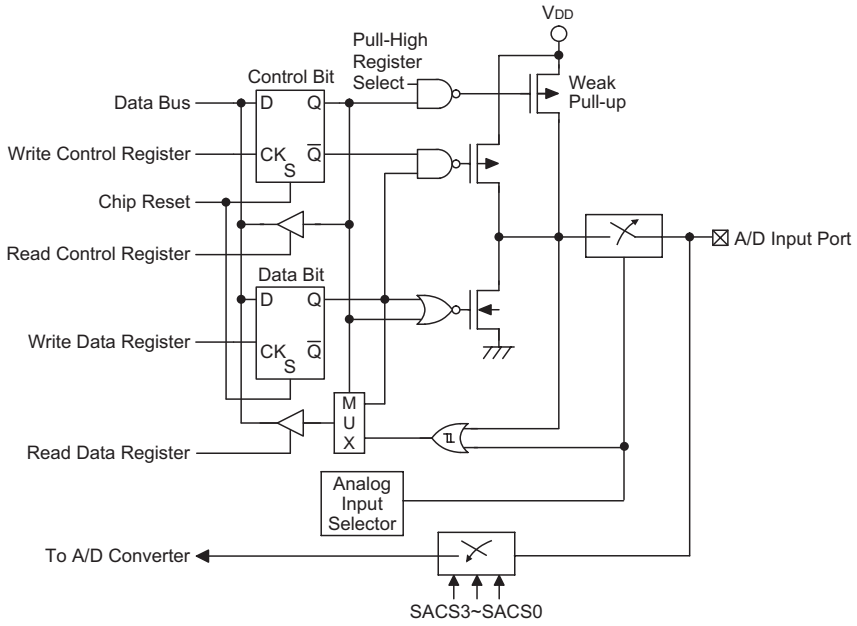
Bit 0 **C0PS**: C0P input source pin selection
0: PC3
1: PD5

I/O Pin Structures

The accompanying diagrams illustrate the internal structures of some generic I/O pin types. As the exact logical construction of the I/O pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins. The wide range of pin-shared structures does not permit all types to be shown.



Generic Input/Output Structure



A/D Input/Output Structure

Programming Considerations

Within the user program, one of the things first to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set to high. This means that all I/O pins will be defaulted 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 functions. 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 the device includes several Timer Modules, generally abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, 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 multiple interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

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

Introduction

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

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

TM Function Summary

Device	CTM	STM	ETM
HT67F60A	10-bit CTM0 10-bit CTM1	16-bit STM0 16-bit STM1 16-bit STM2	10-bit ETM
HT67F70A			

TM 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 count-up counter whose value is then compared with the value of pre-programmed internal comparators. When the free running count-up counter has the same value as the pre-programmed comparator, known as a compare match situation, a TM interrupt signal will be generated which can clear the counter and perhaps also change the condition of the TM output pin. The internal TM counter is driven by a user selectable clock source, which can be an internal clock or an external pin.

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTCK2~xTCK0 bits in the xTM control registers, where "x" can stand for C, S or E and "n" is the serial number. The clock source can be a ratio of the system clock, f_{SYS}, or the internal high clock, f_H, the f_{TBC} clock source or the external xTCK pin. Note that setting these bits to the value 101 will select a reserved clock input, in effect disconnecting the TM clock source. The xTCK pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

TM Interrupts

The Compact or Standard type TM has two internal interrupt, one for each of the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs. As the Enhanced type TM has three internal comparators and comparator A, comparator B or comparator P compare match functions, it consequently has three internal interrupts. 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 or three TM input pins, with the label xTCKn and xTPnI or ETPIA/ETPIB respectively. The TM input pin, xTCKn, is essentially a clock source for the TM 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 TM input pin can be chosen to have either a rising or falling active edge. The STCKn and ETCK pins are also used as the external trigger input pin in single pulse output mode for the STMn and ETM respectively.

The other TM input pin, STPnI or ETPIA/ETPIB, 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 STnIO1~STnIO0 or ETAIO1~ETAIO0/ETBIO1~ETBIO0 bits in the STMnC1 or ETMC1/ ETMC2 register respectively.

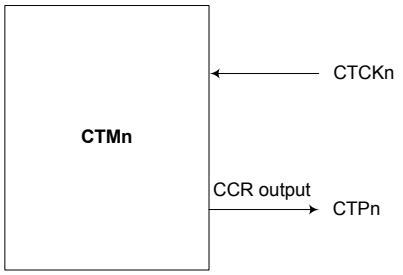
The TMs each have one or more output pins which is selected using the corresponding pin-shared function selection bits described in the Pin-shared Function section. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when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

Type	CTM	STM	ETM	Pin Control Registers
Input	CTCK0, CTCK1	STCK0, STCK1, STCK2 STP0I, STP1I, STP2I	ETCK ETPIA, ETPIB	IFSi
Output	CTP0, CTP1	STP0, STP1, STP2	ETPA, ETPB	PxSn

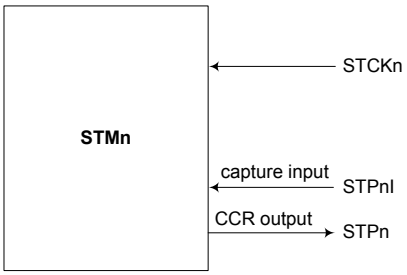
TM External Pins

TM Input/Output Pin Control Register

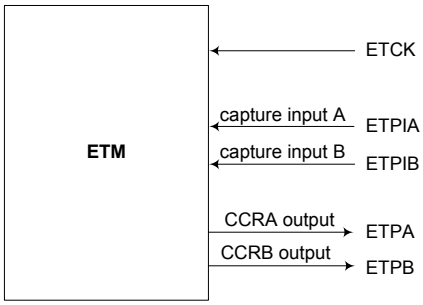
Selecting to have a TM input/output or whether to retain its other shared function is implemented using the relevant pin-shared function selection registers, with the corresponding selection bits in each pin-shared function register corresponding to a TM input/output pin. Configuring the selection bits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



CTMn Functional Pin Diagram (n=0~1)



STMn Functional Pin Diagram (n=0~2)

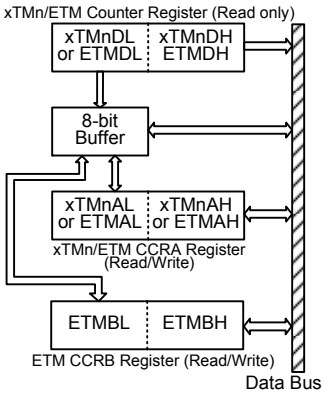


ETM Functional Pin Diagram

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA and CCRB 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 CCRB 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 CCRB low byte registers, named xTMAL or ETMAL and ETMBL, using the following access procedures. Accessing the CCRA or CCRB low byte registers without following these access procedures will result in unpredictable values.



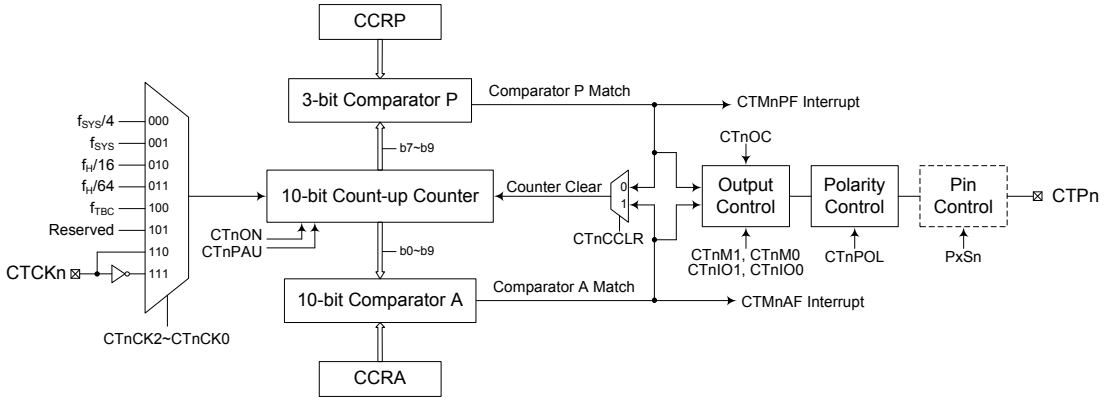
The following steps show the read and write procedures:

- Writing Data to CCRA or CCRB
  - Step 1. Write data to Low Byte xTMnAL, ETMAL or ETMBL
    – note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte xTMnAH, ETMAH or ETMBH
    – here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA or CCRB
  - Step 1. Read data from the High Byte xTMnDH, ETMDH, xTMnAH, ETMAH or ETMBH
    – 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, ETMDL, xTMnAL, ETMAL or ETMBL
    – this step reads data from the 8-bit buffer.

Compact Type TM – CTM

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

Device	TM Type	TM Input Pin	TM Output Pin
HT67F60A/HT67F70A	10-bit CTM (CTM0, CTM1)	CTCK0, CTCK1	CTP0, CTP1



Compact Type TM Block Diagram (n = 0 or 1)

Compact TM Operation

The Compact 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 is three-bit wide whose value is compared with the highest three bits in the counter while the CCRA is ten-bit wide and therefore compares with all counter bits.

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

Compact Type TM Register Description

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

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CTMnC0	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
CTMnC1	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
CTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnDH	—	—	—	—	—	—	D9	D8
CTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnAH	—	—	—	—	—	—	D9	D8

10-bit Compact TM Registers List (n = 0 or 1)

CTMnDL Register

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

Bit 7~0 CTMn Counter Low Byte Register bit 7 ~ bit 0
CTMn 10-bit Counter bit 7 ~ bit 0

CTMnDH Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 CTMn Counter High Byte Register bit 1 ~ bit 0
CTMn 10-bit Counter bit 9 ~ bit 8

CTMnAL Register

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

Bit 7~0 CTMn CCRA Low Byte Register bit 7 ~ bit 0
CTMn 10-bit CCRA bit 7 ~ bit 0

CTMnAH Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 CTMn CCRA High Byte Register bit 1 ~ bit 0
CTMn 10-bit CCRA bit 9 ~ bit 8

CTMnC0 Register

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

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

Bit 6~4 **CTnCK2~CTnCK0**: Select CTMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: Reserved
110: CTCKn rising edge clock
111: CTCKn falling edge clock

These three bits are used to select the clock source for the CTMn. Selecting the Reserved clock input will effectively disable the internal counter. 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_{TBC} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **CTnON**: CTMn Counter On/Off control
0: Off
1: On

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

Bit 2~0 **CTnRP2~CTnRP0**: CTMn CCRP 3-bit register, compared with the CTMn Counter bit9~bit7
000: 1024 CTMn clocks
001: 128 CTMn clocks
010: 256 CTMn clocks
011: 384 CTMn clocks
100: 512 CTMn clocks
101: 640 CTMn clocks
110: 768 CTMn clocks
111: 896 CTMn clocks

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

CTMnC1 Register

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

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

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

Bit 5~4 **CTnIO1~CTnIO0**: Select CTMn external pin (CTPn) function

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

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

Timer/Counter Mode
Unused

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

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

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

Bit 3 **CTnOC**: CTPn Output control

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

PWM Output Mode
0: Active low
1: Active high

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

Bit 2 **CTnPOL**: CTPn Output polarity control
0: Non-inverted
1: Inverted

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

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

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

Bit 0 **CTnCCLR**: CTMn Counter Clear condition selection
0: CTMn Comparator P match
1: CTMn Comparator A match

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

Compact Type TM Operation Modes

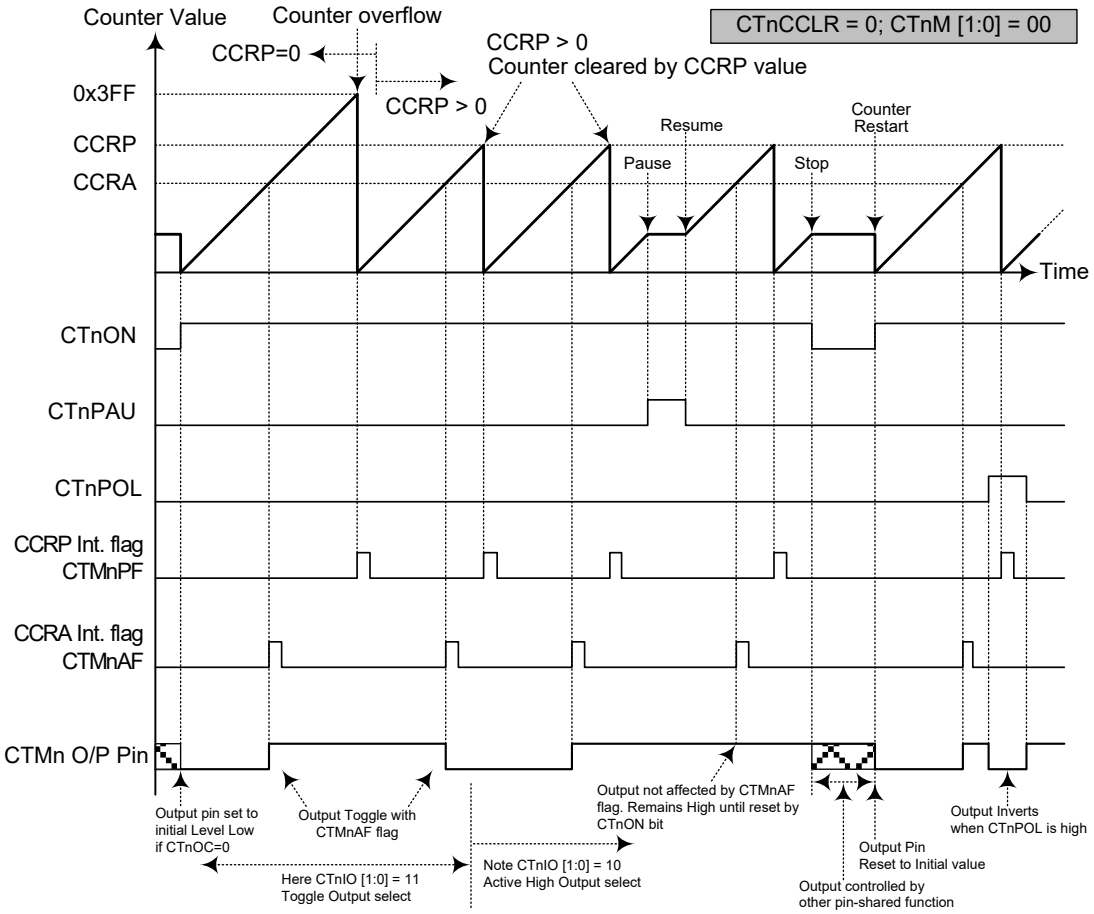
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

Compare Match Output Mode

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

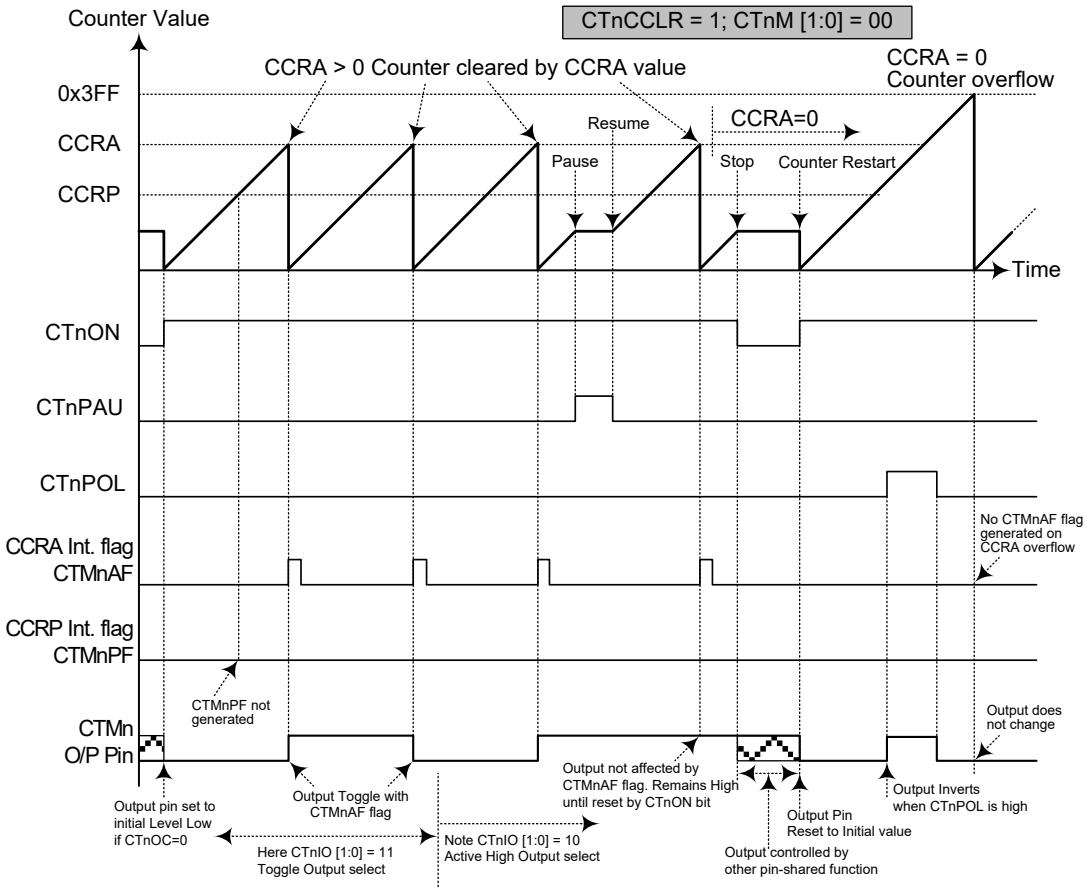
If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is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

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



Compare Match Output Mode – CTnCCLR = 0

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



Compare Match Output Mode – CTnCCLR = 1

Note: 1. With CTnCCLR = 1, a Comparator A match will clear the counter
2. The CTMn output pin is controlled only by CTMnAF flag
3. The CTMn output pin is reset to initial state by CTnON rising edge
4. The CTMnPF flags is not generated when CTnCCLR = 1
5. n = 0 or 1

Timer/Counter Mode

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

PWM Output Mode

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

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

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

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

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

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

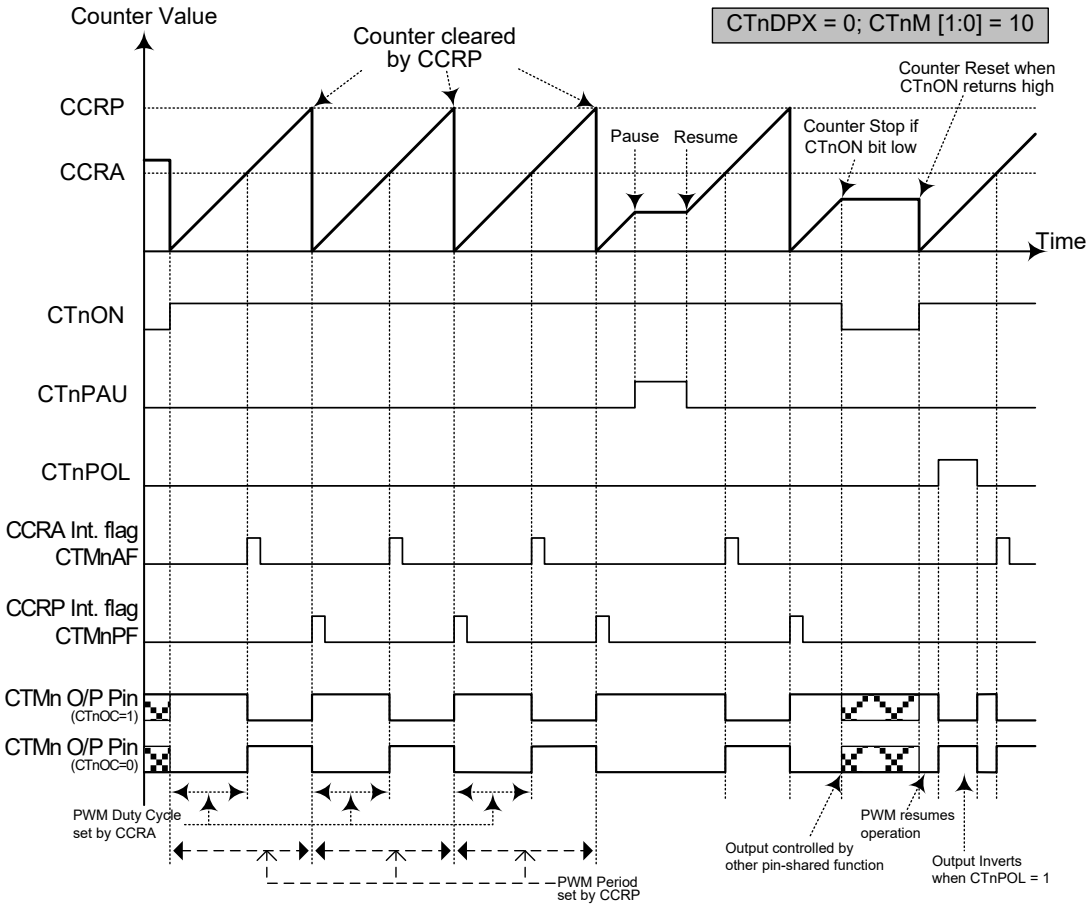
The CTMn PWM output frequency = $(f_{SYS}/4) / (2\times256) = f_{SYS}/2048$ = 7.8125 kHz,
duty = 128/(2×256)= 25%.

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%.

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

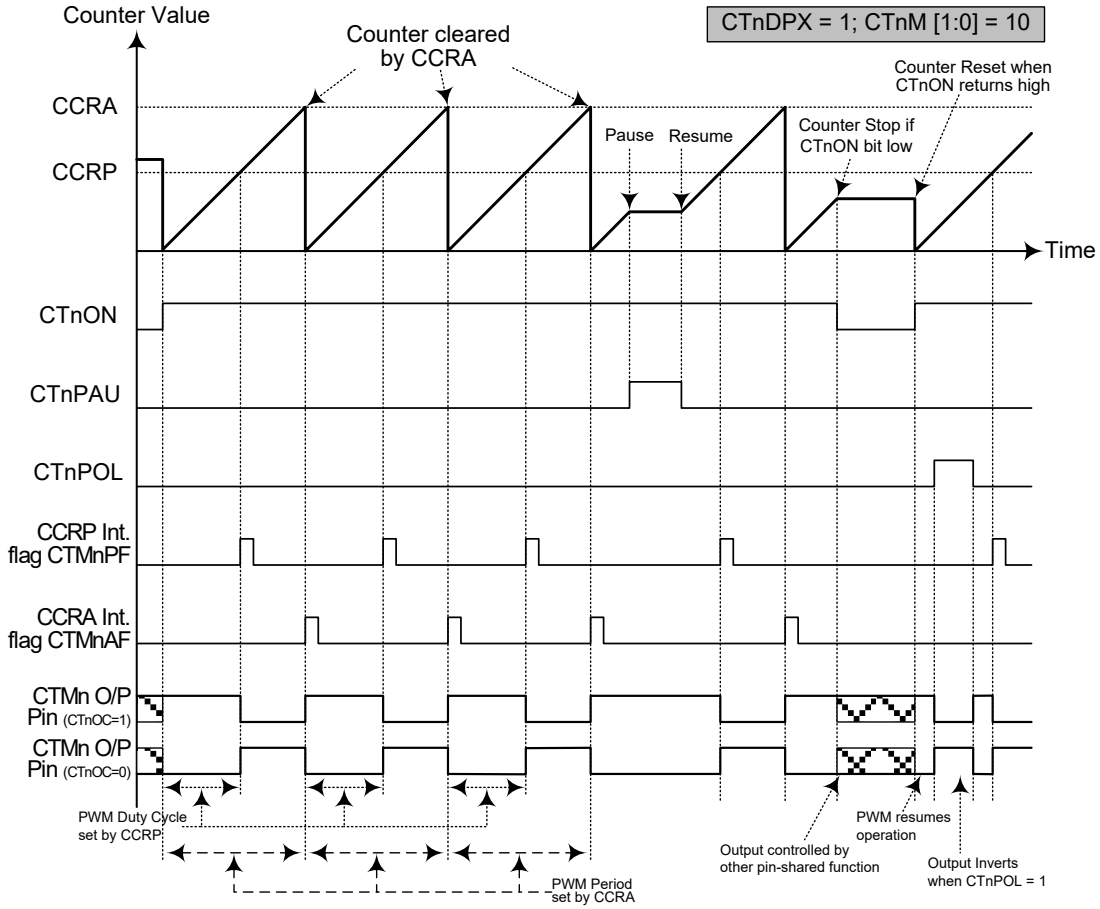
CCRP	001b	011b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

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



PWM Output Mode – CTnDXP = 0

Note: 1. Here CTnDPX = 0 – Counter cleared by CCRP
2. A counter clear sets PWM Period
3. The internal PWM function continues even when CTnIO1, CTnIO0 = 00 or 01
4. The CTnCCLR bit has no influence on PWM operation
5. n = 0 or 1



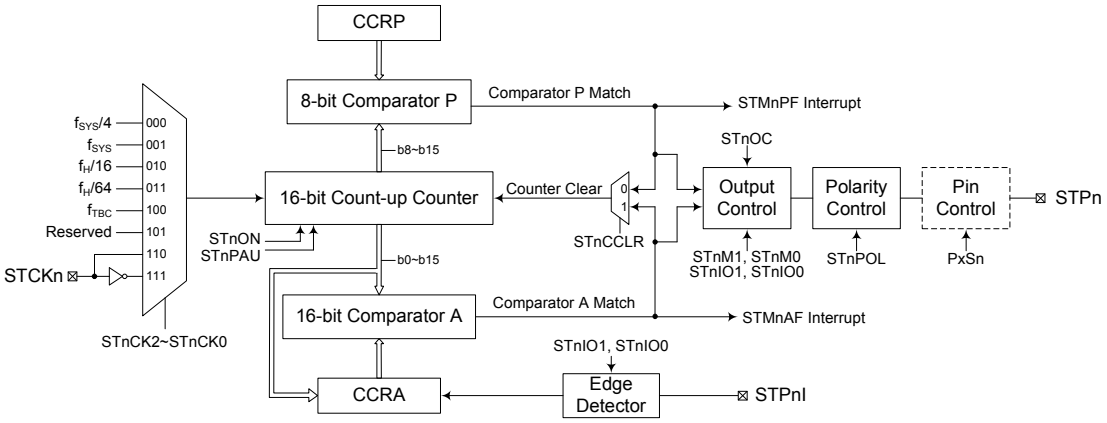
PWM Output Mode – CTnDXP = 1

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

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. Each Standard TM can also be controlled with two external input pins and can drive one external output pin.

Device	TM Type	TM Input Pin	TM Output Pin
HT67F60A HT67F70A	16-bit STM (STM0, STM1, STM2)	STCK0, STCK1, STCK2; STP0I, STP1I, STP2I	STP0, STP1, STP2



Standard Type TM Block Diagram (n = 0, 1 or 2)

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 eight-bit wide whose value is compared with the highest eight bits in the counter while the CCRA is sixteen-bit wide and therefore compares with 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 STnON 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 STMn 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 an output pin. 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 STMnRP register is used to store the 8-bit CCRP bits. The remaining two registers are control registers which setup the different operating and control modes.

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STMnC0	STnPAU	STnCK2	STnCK1	STnCK0	STnON	—	—	—
STMnC1	STnM1	STnM0	STnIO1	STnIO0	STnOC	STnPOL	STnDPX	STnCCLR
STMnDL	D7	D6	D5	D4	D3	D2	D1	D0
STMnDH	D15	D14	D13	D12	D11	D10	D9	D8
STMnAL	D7	D6	D5	D4	D3	D2	D1	D0
STMnAH	D15	D14	D13	D12	D11	D10	D9	D8
STMnRP	STnRP7	STnRP6	STnRP5	STnRP4	STnRP3	STnRP2	STnRP1	STnRP0

16-bit Standard TM Registers List (n = 0, 1 or 2)

STMnDL 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 STMn Counter Low Byte Register bit 7 ~ bit 0
STMn 16-bit Counter bit 7 ~ bit 0

STMnDH 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 STMn Counter High Byte Register bit 7 ~ bit 0
STMn 16-bit Counter bit 15 ~ bit 8

STMnAL 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 STMn CCRA Low Byte Register bit 7 ~ bit 0
STMn 16-bit CCRA bit 7 ~ bit 0

STMnAH 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 STMn CCRA High Byte Register bit 7 ~ bit 0
STMn 16-bit CCRA bit 15 ~ bit 8

STMnC0 Register

Bit	7	6	5	4	3	2	1	0
Name	STnPAU	STnCK2	STnCK1	STnCK0	STnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **STnPAU**: STMn 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 STMn 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 **STnCK2~STnCK0**: Select STMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: Reserved
110: STCKn rising edge clock
111: STCKn falling edge clock

These three bits are used to select the clock source for the STMn. Selecting the Reserved clock input will effectively disable the internal counter. 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_{TBC} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **STnON**: STMn Counter On/Off control
0: Off
1: On

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

Bit 2~0 Unimplemented, read as “0”

STMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	STnM1	STnM0	STnIO1	STnIO0	STnOC	STnPOL	STnDPX	CTnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **STnM1~STnM0**: Select STMn Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the STMn. To ensure reliable operation the STMn should be switched off before any changes are made to the STnM1 and STnM0 bits. In the Timer/Counter Mode, the STMn output pin control will be disabled.

Bit 5~4 **STnIO1~STnIO0**: Select STMn external pin (STPn or STPnI) 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 STPnI
01: Input capture at falling edge of STPnI
10: Input capture at rising/falling edge of STPnI
11: Input capture disabled

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the STnIO1 and STnIO0 bits determine how the STMn output pin changes state when a compare match occurs from the Comparator A. The STMn 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 STMn output pin should be setup using the STnOC bit in the STMnC1 register. Note that the output level requested by the STnIO1 and STnIO0 bits must be different from the initial value setup using the STnOC bit otherwise no change will occur on the STMn output pin when a compare match occurs. After the STMn output pin changes state, it can be reset to its initial level by changing the level of the STnON bit from low to high.

In the PWM Mode, the STnIO1 and STnIO0 bits determine how the STMn 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 STnIO1 and STnIO0 bits only after the STMn has been switched off. Unpredictable PWM outputs will occur if the STnIO1 and STnIO0 bits are changed when the STMn is running.

Bit 3 **STnOC**: STPn 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 STMn output pin. Its operation depends upon whether STMn is being used in the Compare Match Output Mode or in the PWM Mode/Single Pulse Output Mode. It has no effect if the STMn is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of he STMn output pin before a compare match occurs. In the PWM Mode/Single Pulse Output Mode it determines if the PWM signal is active high or active low.

Bit 2 **STnPOL**: STPn Output polarity control
0: Non-inverted
1: Inverted

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

Bit 1 **STnDPX**: STMn 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 **STnCCLR**: STMn 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 STnCCLR 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 STnCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

STMnRP Register

Bit	7	6	5	4	3	2	1	0
Name	STnRP7	STnRP6	STnRP5	STnRP4	STnRP3	STnRP2	STnRP1	STnRP0
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 **STnRP7~STnRP0**: STMn CCRP 8-bit register, compared with the STMn counter bit15~bit8

Comparator P match period =
0: 65536 STMn clocks
1~255: (1~255) × 256 STMn 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 STnCCLR bit is set to zero. Setting the STnCCLR 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

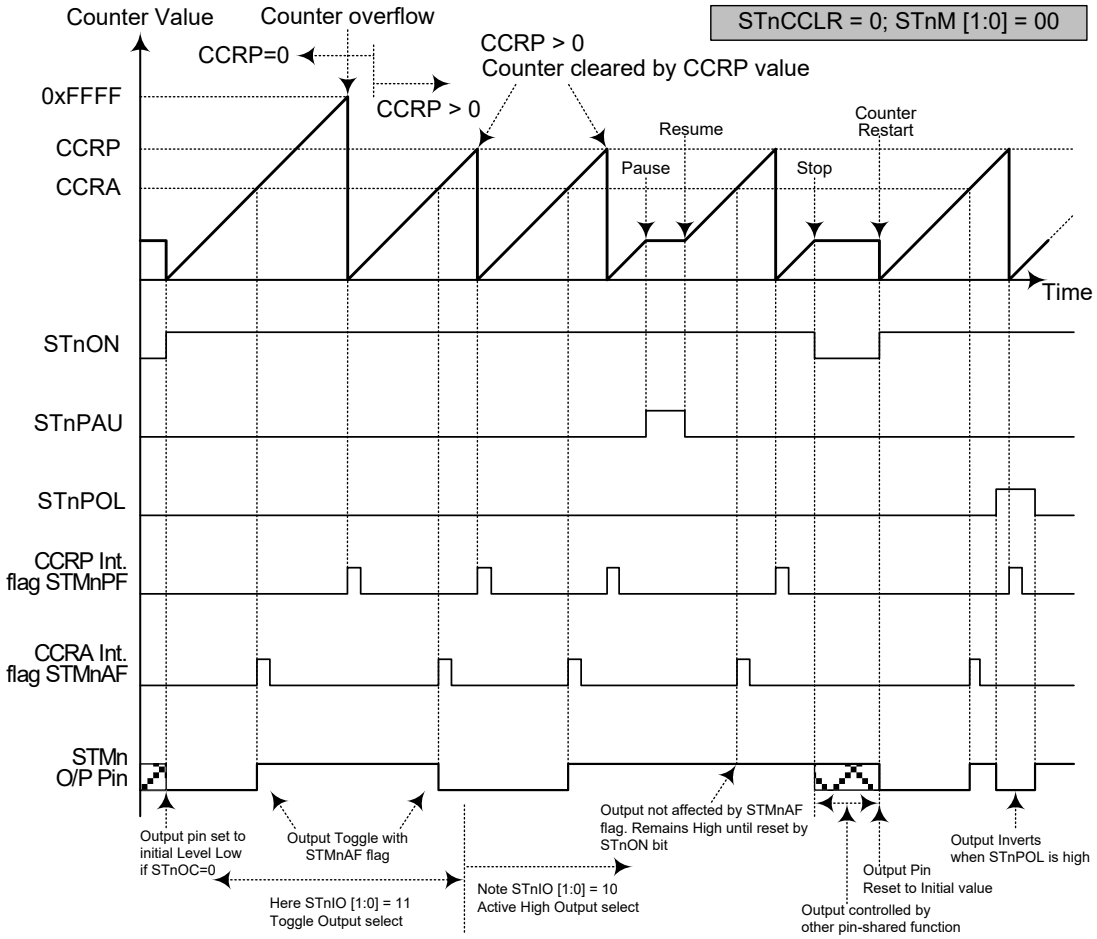
The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STnM1 and STnM0 bits in the STMnC1 register.

Compare Match Output Mode

To select this mode, bits STnM1 and STnM0 in the STMnC1 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 STnCCLR 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 STMnAF and STMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

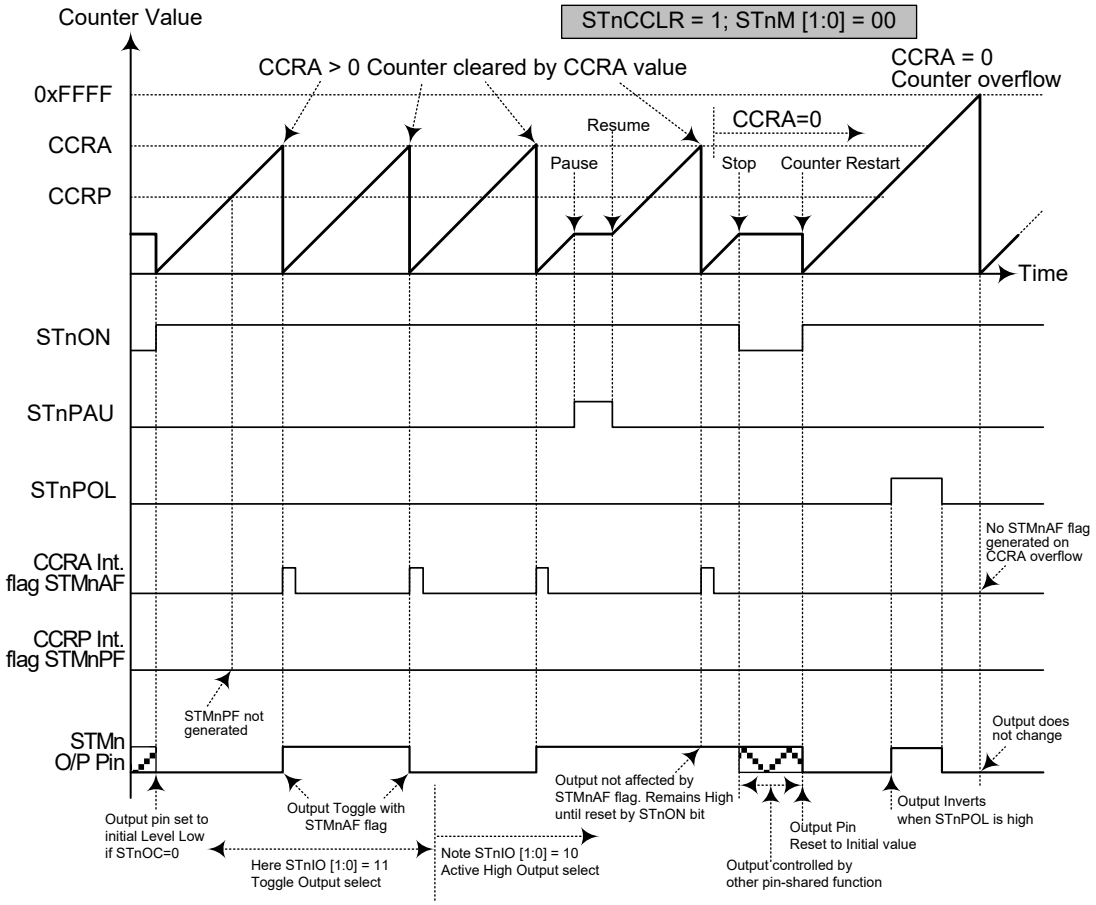
If the STnCCLR bit in the STMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when STnCCLR is high no STMnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”.

As the name of the mode suggests, after a comparison is made, the STMn output pin, will change state. The STMn output pin condition however only changes state when a STMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The STMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the STMn output pin. The way in which the STMn output pin changes state are determined by the condition of the STnIO1 and STnIO0 bits in the STMnC1 register. The STMn output pin can be selected using the STnIO1 and STnIO0 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 STMn output pin, which is setup after the STnON bit changes from low to high, is setup using the STnOC bit. Note that if the STnIO1 and STnIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – STnCCLR = 0

Note: 1. With STnCCLR=0 a Comparator P match will clear the counter
2. The STMn output pin is controlled only by the STMnAF flag
3. The output pin is reset to itsinitial state by a STnON bit rising edge
4. n = 0, 1 or 2



Compare Match Output Mode – STnCCLR = 1

Note:1. With STnCCLR=1 a Comparator A match will clear the counter
2. The STMn output pin is controlled only by the STMnAF flag
3. The output pin is reset to its initial state by a STnON bit rising edge
4. A STMnPF flag is not generated when STnCCLR=1
5. n = 0, 1 or 2

Timer/Counter Mode

To select this mode, bits STnM1 and STnM0 in the STMnC1 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 STMn 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 STMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits STnM1 and STnM0 in the STMnC1 register should be set to 10 respectively and also the STnIO1 and STnIO0 bits should be set to 10 respectively. The PWM function within the STMn 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 STMn 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 mode, the STnCCLR 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 STnDPX bit in the STMnC1 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 STnOC bit in the STMnC1 register is used to select the required polarity of the PWM waveform while the two STnIO1 and STnIO0 bits are used to enable the PWM output or to force the STMn output pin to a fixed high or low level. The STnPOL bit is used to reverse the polarity of the PWM output waveform.

- **16-bit STMn, PWM Mode, Edge-aligned Mode, STnDPX=0**

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

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

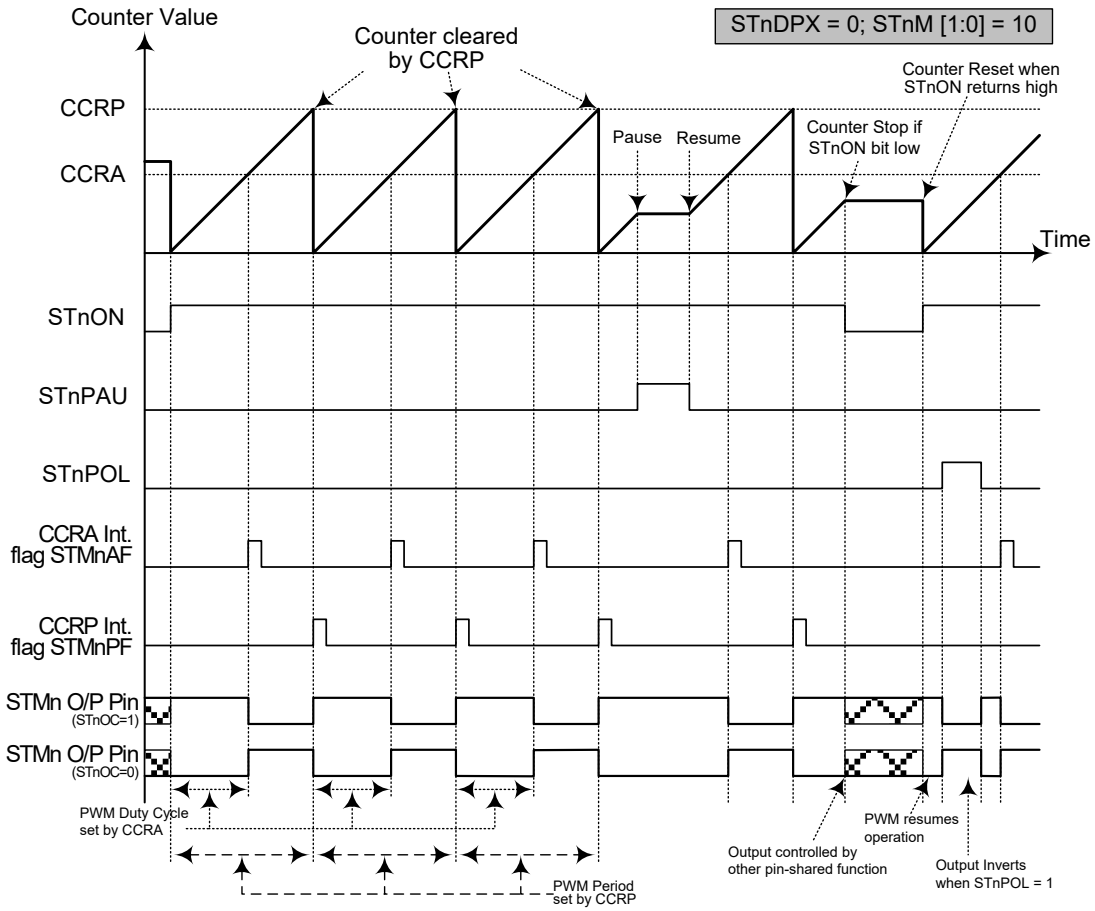
The STMn PWM output frequency = ($f_{SYS}/4$) / (2×256) = $f_{SYS}/2048$ = 7.8125 kHz,
duty = 128/(2×256)= 25%.

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 STMn, PWM Mode, Edge-aligned Mode, STnDPX=1**

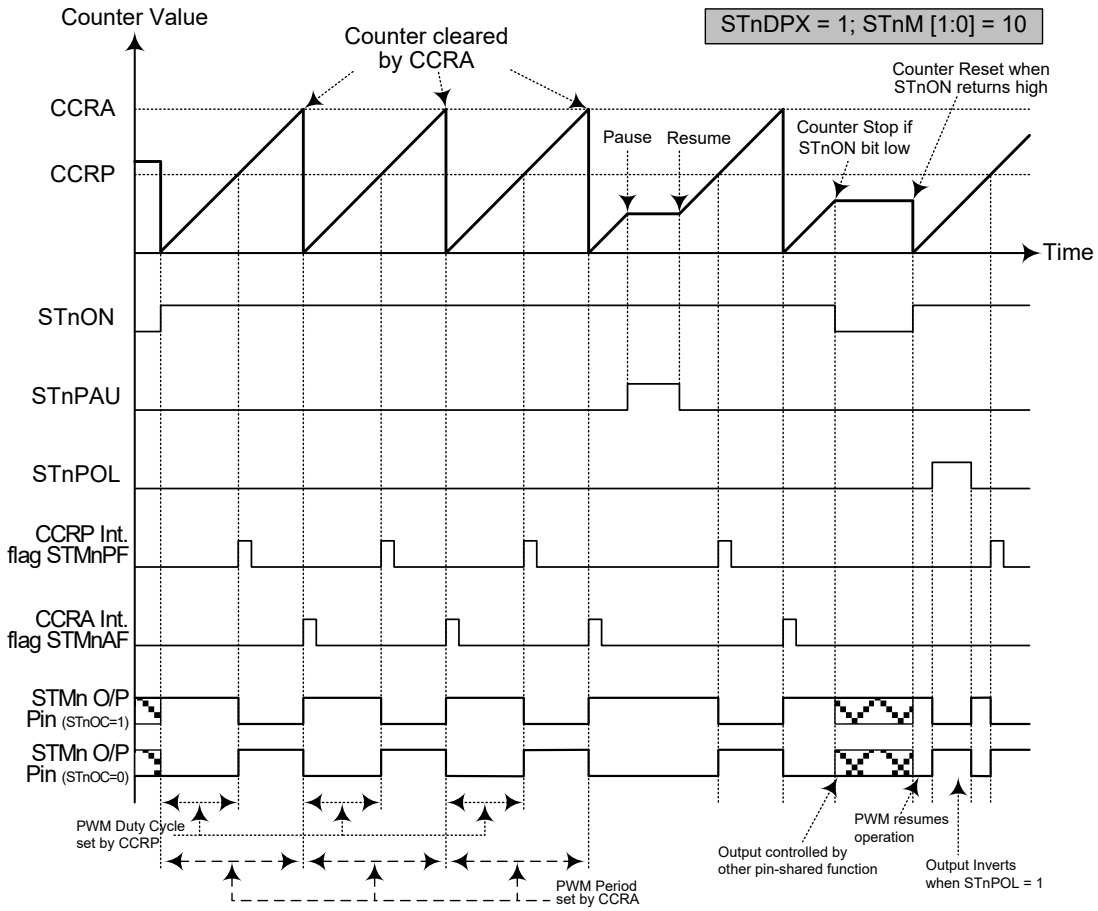
CCRP	1~255	0
Period	CCRA	CCRA
Duty	CCRP×256	65536

The PWM output period is determined by the CCRA register value together with the STMn 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 – STnDXP = 0

Note: 1. Here STnDPX=0 – Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when STnIO [1:0] = 00 or 01
4. The STnCCLR bit has no influence on PWM operation
5. n = 0, 1 or 2



PWM Output Mode – STnDXP = 1

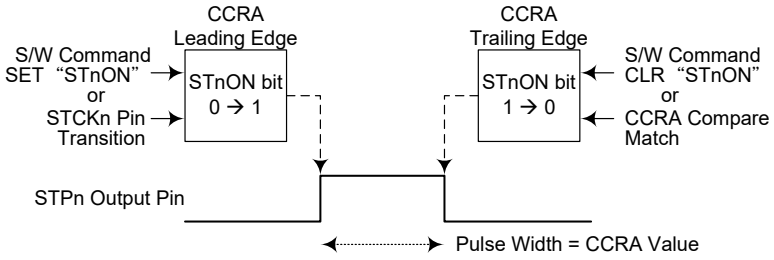
Note: 1. Here STnDPX=1 – Counter cleared by CCRA
2. A counter clear sets the PWM Period
3. The internal PWM function continues even when STnIO [1:0] = 00 or 01
4. The STnCCLR bit has no influence on PWM operation
5. n = 0, 1 or 2

Single Pulse Output Mode

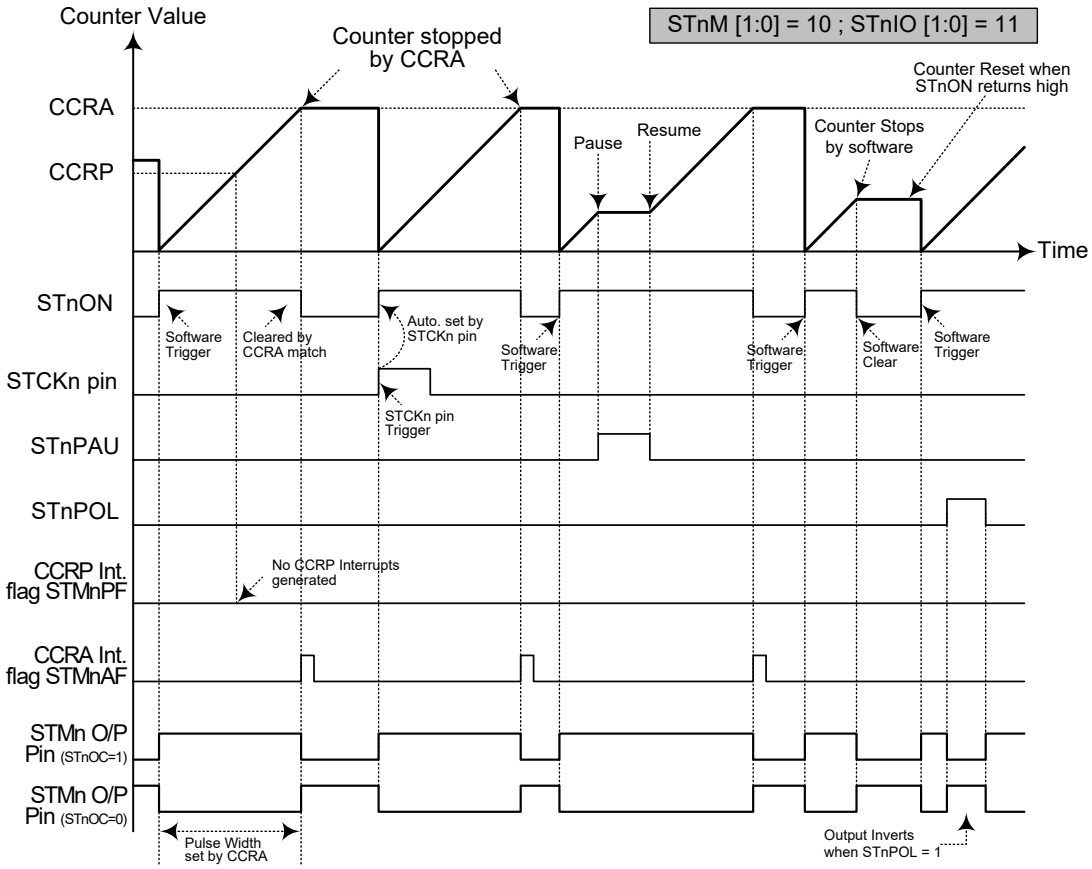
To select this mode, bits STnM1 and STnM0 in the STMnC1 register should be set to 10 respectively and also the STnIO1 and STnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STMn output pin.

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



Single Pulse Generation



Single Pulse Output Mode

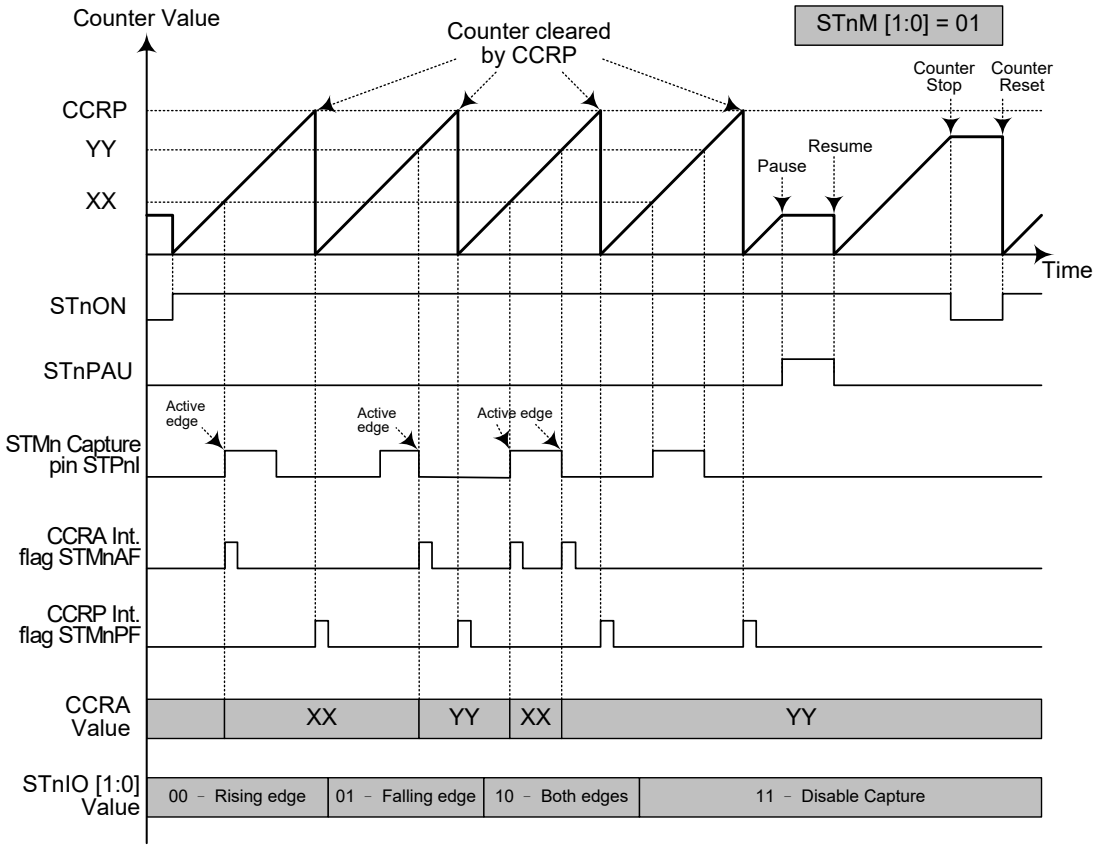
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse triggered by the STCKn pin or by setting the STnON bit high
4. A STCKn pin active edge will automatically set the STnON bit high.
5. In the Single Pulse Mode, STnIO [1:0] must be set to “11” and can not be changed.
6. n = 0, 1 or 2

Capture Input Mode

To select this mode bits STnM1 and STnM0 in the STMnC1 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 STPnI 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 STnIO1 and STnIO0 bits in the STMnC1 register. The counter is started when the STnON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the STPnI pin the present value in the counter will be latched into the CCRA registers and a STMn interrupt generated. Irrespective of what events occur on the STPnI pin the counter will continue to free run until the STnON 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 STMn 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 STnIO1 and STnIO0 bits can select the active trigger edge on the STPnI pin to be a rising edge, falling edge or both edge types. If the STnIO1 and STnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the STPnI pin, however it must be noted that the counter will continue to run.

The STnCCLR and STnDPX bits are not used in this Mode.



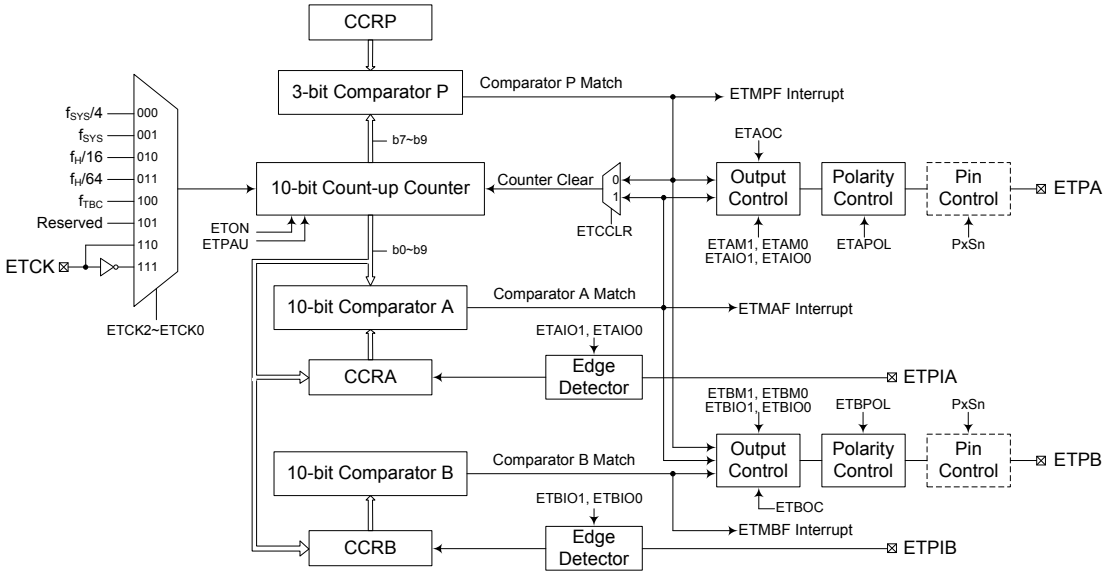
Capture Input Mode

Note: 1. STnM [1:0] = 01 and active edge set by the STnIO [1:0] bits
2. A STMn Capture input pin active edge transfers the counter value to CCRA
3. STnCCLR bit not used
4. No output function – STnOC and STnPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero.
6. n = 0, 1 or 2

Enhanced Type TM – ETM

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

Device	TM Type	TM Input Pin	TM Output Pin
HT67F60A/HT67F70A	10-bit ETM (ETM)	ETCK; ETPIA, ETPIB	ETPA, ETPB



Enhanced Type TM Block Diagram

Enhanced TM Operation

At its core is a 10-bit count-up/count-down counter which is driven by a user selectable internal or external clock source. There are three internal comparators with the names, Comparator A, Comparator B and Comparator P. These comparators will compare the value in the counter with the CCRA, CCRB and CCRP registers. The CCRP comparator is 3-bit wide whose value is compared with the highest 3 bits in the counter while CCRA and CCRB are 10-bit wide and therefore 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 ETON 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 ETM interrupt signal will also usually be generated. The Enhanced 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 output pins. All operating setup conditions are selected using relevant internal registers.

Enhance Type TM Register Description

Overall operation of the Enhanced 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 and CCRB value. The remaining three two registers are control registers which setup the different operating and control modes as well as the three CCRP bits.

Register Name	Bit							
	7	6	5	4	3	2	1	0
ETMC0	ETPAU	ETCK2	ETCK1	ETCK0	ETON	ETRP2	ETRP1	ETRP0
ETMC1	ETAM1	ETAM0	ETAIO1	ETAIO0	ETAOC	ETAPOL	ETCDN	ETCCLR
ETMC2	ETBM1	ETBM0	ETBIO1	ETBIO0	ETBOC	ETBPOL	ETPWM1	ETPWM0
ETMDL	D7	D6	D5	D4	D3	D2	D1	D0
ETMDH	—	—	—	—	—	—	D9	D8
ETMAL	D7	D6	D5	D4	D3	D2	D1	D0
ETMAH	—	—	—	—	—	—	D9	D8
ETMBL	D7	D6	D5	D4	D3	D2	D1	D0
ETMBH	—	—	—	—	—	—	D9	D8

10-bit Enhanced TM Registers List

ETMDL 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 ETM Counter Low Byte Register bit 7 ~ bit 0
ETM 10-bit Counter bit 7 ~ bit 0

ETMDH Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 ETM Counter High Byte Register bit 1 ~ bit 0
ETM 10-bit Counter bit 9 ~ bit 8

ETMAL 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 ETM CCRA Low Byte Register bit 7 ~ bit 0
ETM 10-bit CCRA bit 7 ~ bit 0

ETMAH Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 ETM CCRA High Byte Register bit 1 ~ bit 0
ETM 10-bit CCRA bit 9 ~ bit 8

ETMBL 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 ETM CCRB Low Byte Register bit 7 ~ bit 0
ETM 10-bit CCRB bit 7 ~ bit 0

ETMBH Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 ETM CCRB High Byte Register bit 1 ~ bit 0
ETM 10-bit CCRB bit 9 ~ bit 8

ETMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	ETPAU	ETCK2	ETCK1	ETCK0	ETON	ETRP2	ETRP1	ETRP0
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 **ETPAU**: ETM 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 ETM 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 **ETCK2~ETCK0**: Select ETM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: Reserved
110: ETCK rising edge clock
111: ETCK falling edge clock

These three bits are used to select the clock source for the ETM. Selecting the Reserved clock input will effectively disable the internal counter. 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_{TBC} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **ETON**: ETM Counter On/Off control
0: Off
1: On

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

Bit 2~0 **ETRP2~ETRP0**: ETM CCRP 3-bit register, compared with the ETM Counter bit 9~bit 7
000: 1024 ETM clocks
001: 128 ETM clocks
010: 256 ETM clocks
011: 384 ETM clocks
100: 512 ETM clocks
101: 640 ETM clocks
110: 768 ETM clocks
111: 896 ETM clocks

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

ETMC1 Register

Bit	7	6	5	4	3	2	1	0
Name	ETAM1	ETAM0	ETAIO1	ETAIO0	ETAOC	ETAPOL	ETCDN	ETCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **ETAM1~ETAM0**: Select ETM CCRA Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the ETM CCRA. To ensure reliable operation the ETM should be switched off before any changes are made to the ETAM1 and ETAM0 bits. In the Timer/Counter Mode, the ETM output pin control will be disabled.

Bit 5~4 **ETAIO1~ETAIO0**: Select ETM external pin ETPA or ETPIA 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 ETPIA
01: Input capture at falling edge of ETPIA
10: Input capture at rising/falling edge of ETPIA
11: Input capture disabled

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the ETAIO1 and ETAIO0 bits determine how the ETM ETPA output pin changes state when a compare match occurs from the Comparator A. The ETM ETPA 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 ETM ETPA output pin should be setup using the ETAOC bit in the ETMC1 register. Note that the output level requested by the ETAIO1 and ETAIO0 bits must be different from the initial value setup using the ETAOC bit otherwise no change will occur on the ETM ETPA output pin when a compare match occurs. After the ETM ETPA output pin changes state, it can be reset to its initial level by changing the level of the ETON bit from low to high.

In the PWM Mode, the ETAIO1 and ETAIO0 bits determine how the ETM ETPA 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 ETAIO1 and ETAIO0 bits only after the ETM has been switched off. Unpredictable PWM outputs will occur if the ETAIO1 and ETAIO0 bits are changed when the ETM is running.

Bit 3 **ETAOC**: ETPA 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 ETM ETPA output pin. Its operation depends upon whether ETM 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 ETM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the ETM ETPA output pin before a compare match occurs. In the PWM Output Mode/Single Pulse Output Mode it determines if the PWM signal is active high or active low.

Bit 2 **ETAPOL**: ETPA Output polarity control
0: Non-inverted
1: Inverted

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

Bit 1 **ETCDN**: ETM Counter count up or down flag
0: Count up
1: Count down

Bit 0 **ETCCLR**: ETM Counter Clear condition selection
0: ETM Comparator P match
1: ETM Comparator A match

This bit is used to select the method which clears the counter. Remember that the Enhanced TM contains three comparators, Comparator A, Comparator B and Comparator P, but only Comparator A or Comparator P can be selected to clear the internal counter. With the ETCCLR 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 ETCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

ETMC2 Register

Bit	7	6	5	4	3	2	1	0
Name	ETBM1	ETBM0	ETBIO1	ETBIO0	ETBOC	ETBPOL	ETPWM1	ETPWM0
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 **ETBM1~ETBM0**: Select ETM CCRB Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the ETM CCRB. To ensure reliable operation the ETM should be switched off before any changes are made to the ETBM1 and ETBM0 bits. In the Timer/Counter Mode, the ETM output pin control will be disabled.

Bit 5~4 **ETBIO1~ETBIO0**: Select ETM external pin ETPB or ETPIB 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 ETPIB
01: Input capture at falling edge of ETPIB
10: Input capture at rising/falling edge of ETPIB
11: Input capture disabled

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the ETBIO1 and ETBIO0 bits determine how the ETM ETPB output pin changes state when a compare match occurs from the Comparator A. The ETM ETPB 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 ETM ETPB output pin should be setup using the ETBOC bit in the ETMC2 register. Note that the output level requested by the ETBIO1 and ETBIO0 bits must be different from the initial value setup using the ETBOC bit otherwise no change will occur on the ETM ETPB output pin when a compare match occurs. After the ETM ETPB output pin changes state, it can be reset to its initial level by changing the level of the ETON bit from low to high.

In the PWM Mode, the ETBIO1 and ETBIO0 bits determine how the ETM ETPB 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 ETBIO1 and ETBIO0 bits only after the ETM has been switched off. Unpredictable PWM outputs will occur if the ETBIO1 and ETBIO0 bits are changed when the ETM is running.

Bit 3 **ETBOC**: ETPB 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 ETM ETPB output pin. Its operation depends upon whether ETM 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 ETM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the ETM ETPB output pin before a compare match occurs. In the PWM Output Mode/Single Pulse Output Mode it determines if the PWM signal is active high or active low.

Bit 2 **ETBPOL**: ETPB Output polarity control
0: Non-inverted
1: Inverted

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

Bit 1~0 **ETPWM1~ETPWM0**: Select ETM PWM Mode
00: Edge aligned
01: Centre aligned, compare match on counter count up
10: Centre aligned, compare match on counter count down
11: Centre aligned, compare match on counter count up or down

Enhanced Type TM Operation Modes

The Enhanced 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 ETAM1 and ETAM0 bits in the ETMC1 register and the ETBM1 and ETBM0 bits in the ETMC2 register

ETM Operation Mode	CCRA Compare Match Output Mode	CCRA Timer/ Counter Mode	CCRA PWM Output Mode	CCRA Single Pulse Output Mode	CCRA Capture Input Mode
CCRB Compare Match Output Mode	√	—	—	—	—
CCRB Timer/ Counter Mode	—	√	—	—	—
CCRB PWM Output Mode	—	—	√	—	—
CCRB Single Pulse Output Mode	—	—	—	√	—
CCRB Capture Input Mode	—	—	—	—	√

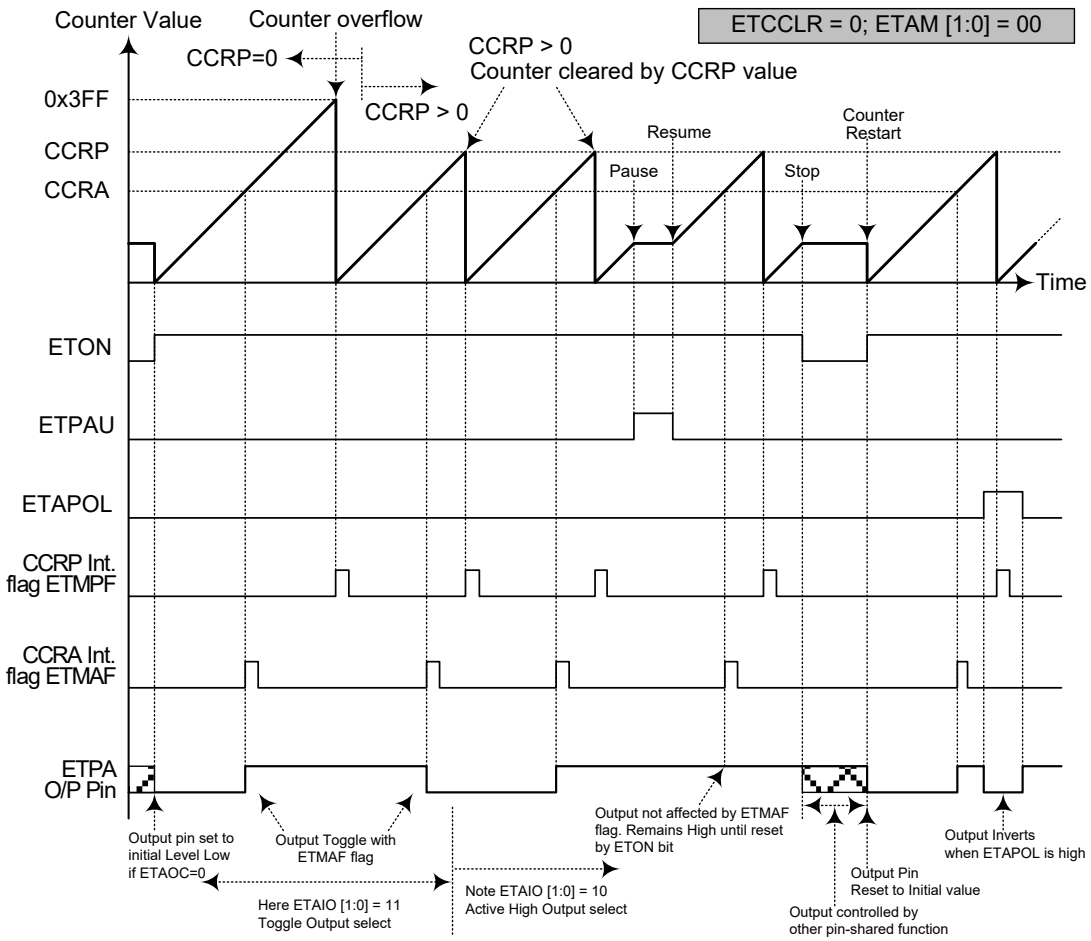
"√": permitted; "—": not permitted

Compare Match Output Mode

To select this mode, bits ETAM1, ETAM0, ETBM1 and ETBM0 in the ETMC1 and ETMC2 registers should be all cleared 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 ETCCLR 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 ETMAF and ETMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

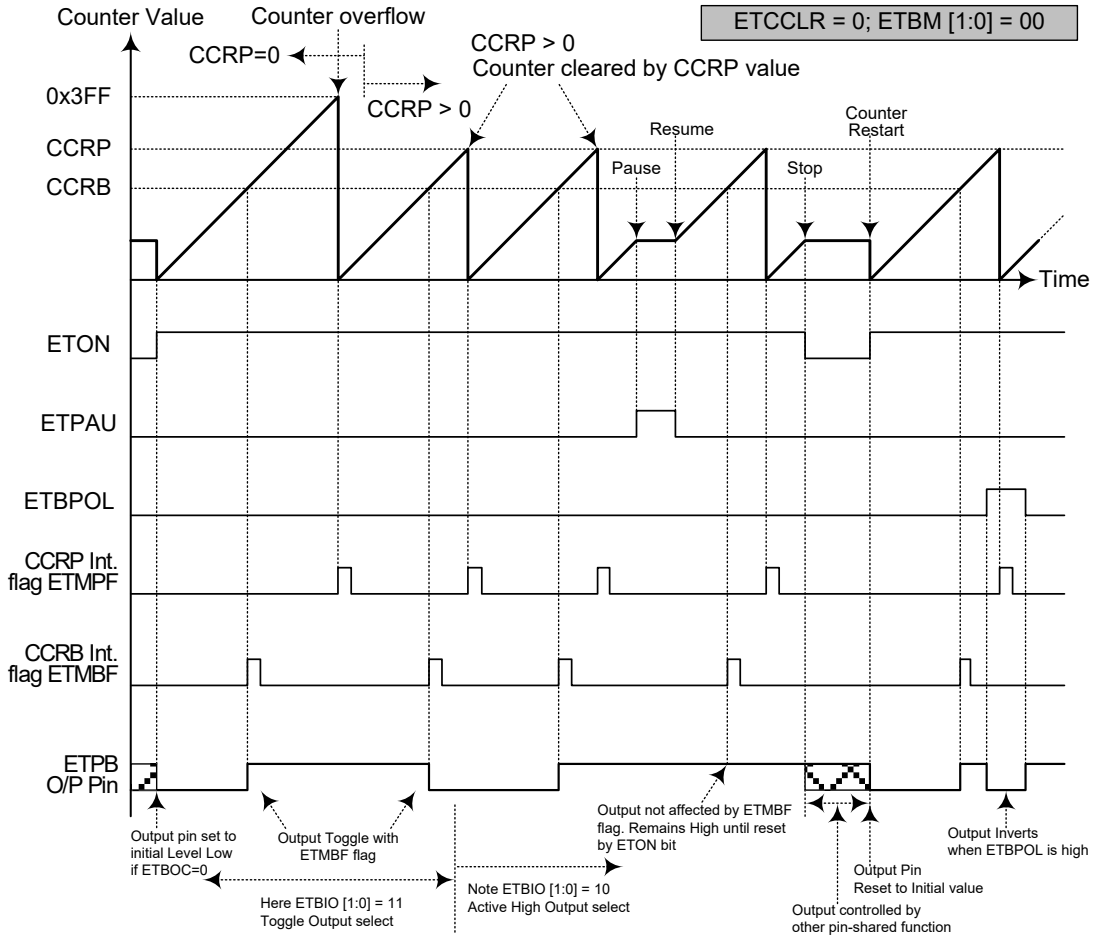
If the ETCCLR bit in the ETMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the ETMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when ETCCLR is high no ETMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”.

As the name of the mode suggests, after a comparison is made, the ETM output pin will change state. The ETM output pin condition however only changes state when a ETMAF or ETMBF interrupt request flag is generated after a compare match occurs from Comparator A or Comparator B. The ETMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the ETM output pin. The way in which the ETM output pin changes state are determined by the condition of the ETAIO1 and ETAIO0 bits in the ETMC1 register for ETM CCRA and the condition of the ETBIO1 and ETBIO0 bits in the ETMC2 register for ETM CCRB . The ETM ETPA or ETPB output pin can be selected using the ETAIO1 and ETAIO0 or ETBIO1 and ETBIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A or Comparator B respectively. The initial condition of the ETM ETPA or ETPB output pin, which is setup after the ETON bit changes from low to high, is setup using the ETAOC or ETBOC bit. Note that if the ETAIO1, ETAIO0, ETBIO1 and ETBIO0 bits are zero then no pin change will take place.



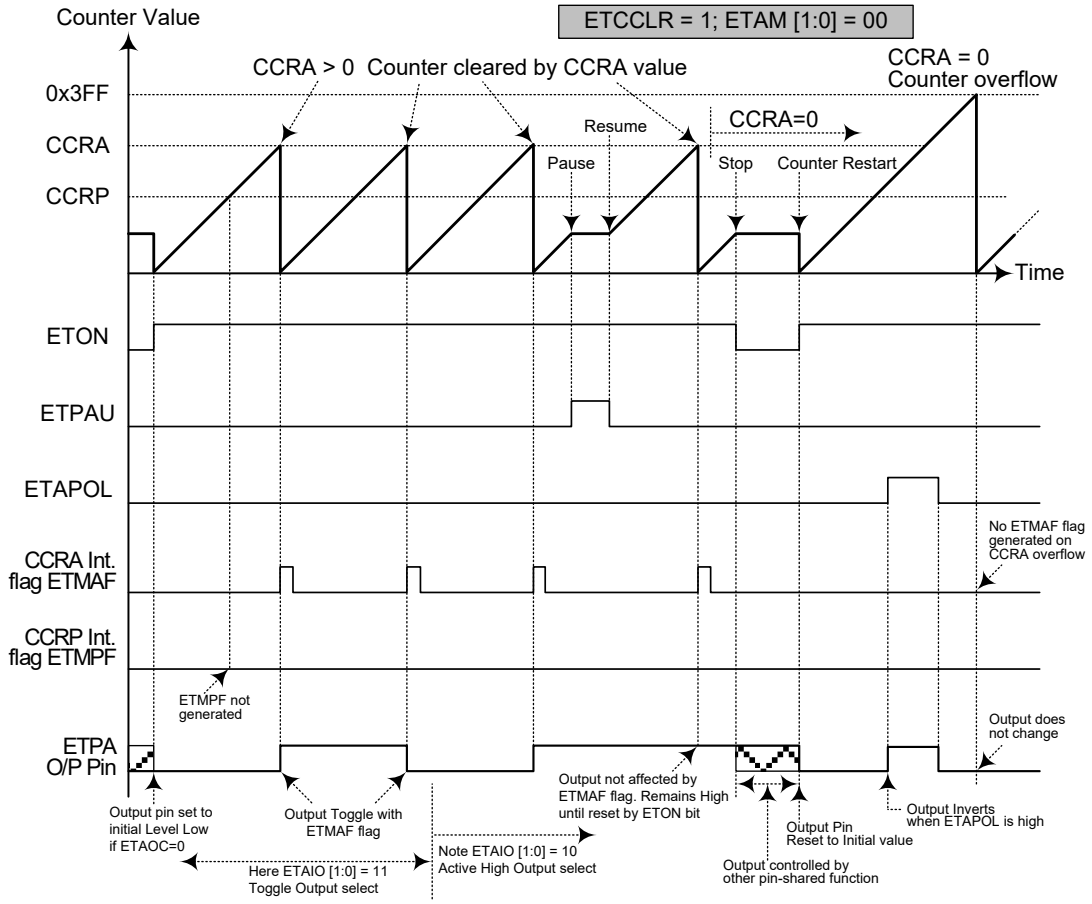
10-bit ETM CCRA Compare Match Output Mode – ETCCLR = 0

Note: 1. With ETCCLR=0, a Comparator P match will clear the counter
2. The ETM ETPA output pin is controlled only by the ETMAF flag
3. The output pin is reset to its initial state by a ETON bit rising edge



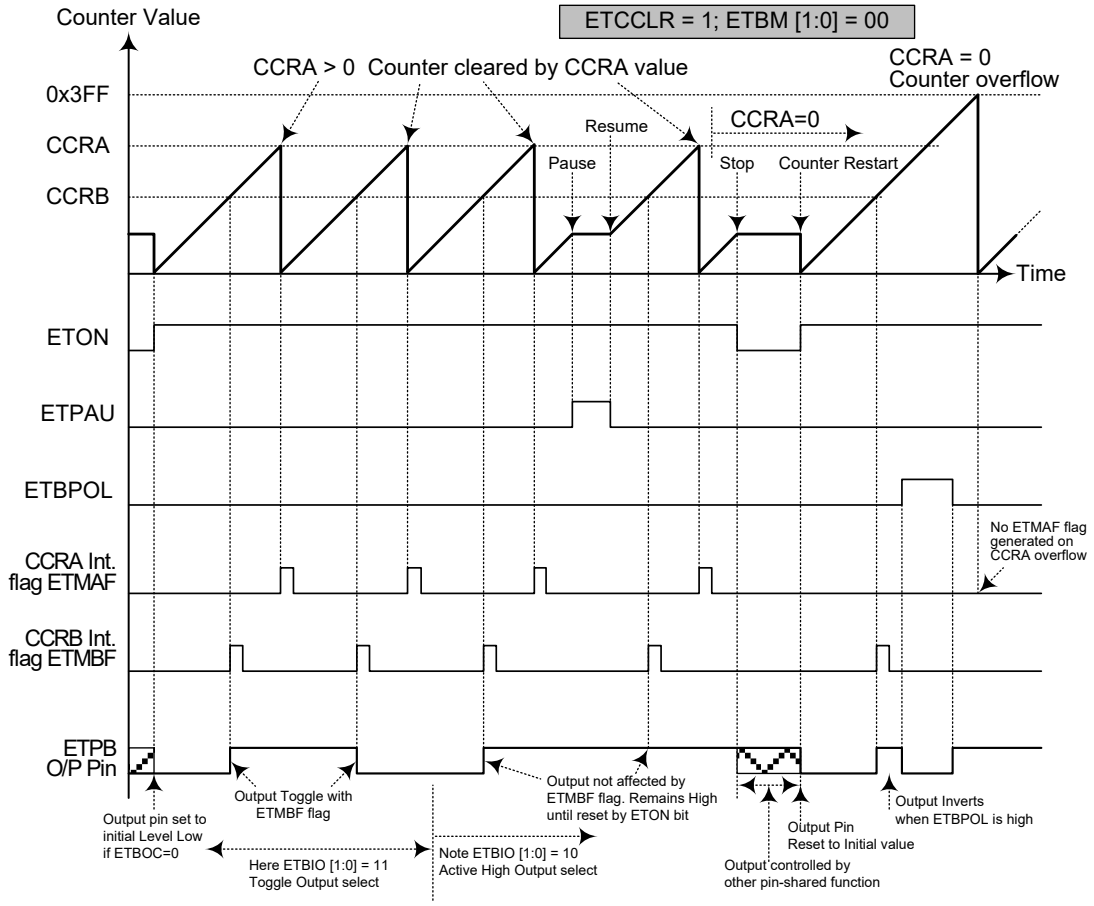
10-bit ETM CCRB Compare Match Output Mode – ETCCLR = 0

Note: 1. With ETCCLR=0, a Comparator P match will clear the counter
2. The ETM ETPB output pin is controlled only by the ETMBF flag
3. The output pin is reset to its initial state by a ETON bit rising edge



10-bit ETM CCRA Compare Match Output Mode – ETCCLR = 1

Note: 1. With ETCCLR=1, a Comparator A match will clear the counter
2. The ETM ETPA output pin is controlled only by the ETMAF flag
3. The output pin is reset to its initial state by a ETON bit rising edge
4. A ETMPF flag is not generated when ETCCLR =1



10-bit ETM CCRB Compare Match Output Mode – ETCCLR = 1

Note: 1. With ETCCLR=1, a Comparator A match will clear the counter
2. The ETM ETPB output pin is controlled only by the ETMBF flag
3. The output pin is reset to its initial state by a ETON bit rising edge
4. A ETMPF flag is not generated when ETCCLR =1

Timer/Counter Mode

To select this mode, bits ETAM1, ETAM0, ETBM1 and ETBM0 in the ETMC1 and ETMC2 registers 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 ETM 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 ETM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, the required bit pairs, ETAM1, ETAM0 and ETBM1, ETBM0 in the ETMC1 and ETMC2 register should be set to 10 respectively and also the ETAIO1, ETAIO0 and ETBIO1, ETBIO0 bits should be set to 10 respectively. The PWM function within the ETM 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 ETM 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 mode, the ETCCLR bit is used to determine in which way the PWM period is controlled. With the ETCCLR bit is high, the PWM period can be finely controlled using the CCRA registers. In this case the CCRB registers are used to set the PWM duty valye for ETPB output pin. The CCRP bits are not used and ETPA output pin is not used. The PWM output can only be generated on the ETPB output pin. With the ETCCLR bit cleared to zero, the PWM period is set using one of the eight values of the three CCRP bits, in multiples of 128. Now both CCRA and CCRB registers can be used to setup different duty cycle values to provide dual PWM outputs on their relative ETPA and ETPB pins.

The ETPWM1 and ETPWM0 bits determine the PWM alignment type, which can be either edge or centre type. In edge alignment, the leading edge of the PWM signals will all be generated concurrently when the counter is reset to zero. With all power currents switching on at the same time, this may give rise to problems in higher power applications. In centre alignment the centre of the PWM active signals will occur sequentially, this reducing the level of simultaneous power switching currents.

Interrupt flags, one for each of the CCRA, CCRB and CCRP, will be generated when a compare match occurs from Comparator A, Comparator B or Comparator P. The ETAOC and ETBOC bits in the ETMC1 and ETMC2 registers are used to select the required polarity of the PWM waveform while the two ETAIO1, ETAIO0 and ETBIO1, ETBIO0 bit pairs are used to enable the PWM output or to force the ETM output pin to a fixed high or low level. The ETAPOL and ETBPOL bits are used to reverse the polarity of the PWM output waveform.

- **10-bit ETM, PWM Mode, Edge-aligned Mode, ETCCLR=0**

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

If f_{SYS}=12MHz, ETM clock source select f_{SYS}/4, CCRP=100b and CCRA=128 and CCRB=256,

The ETPA PWM output frequency = (f_{SYS}/4)/512 = f_{SYS}/2048 = 5.8594kHz, duty=128/512=25%.

The ETPB PWM output frequency = (f_{SYS}/4)/512 = f_{SYS}/2048 = 5.8594kHz, duty=256/512=50%.

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

- **10-bit ETM, PWM Mode, Edge-aligned Mode, ETCCLR=1**

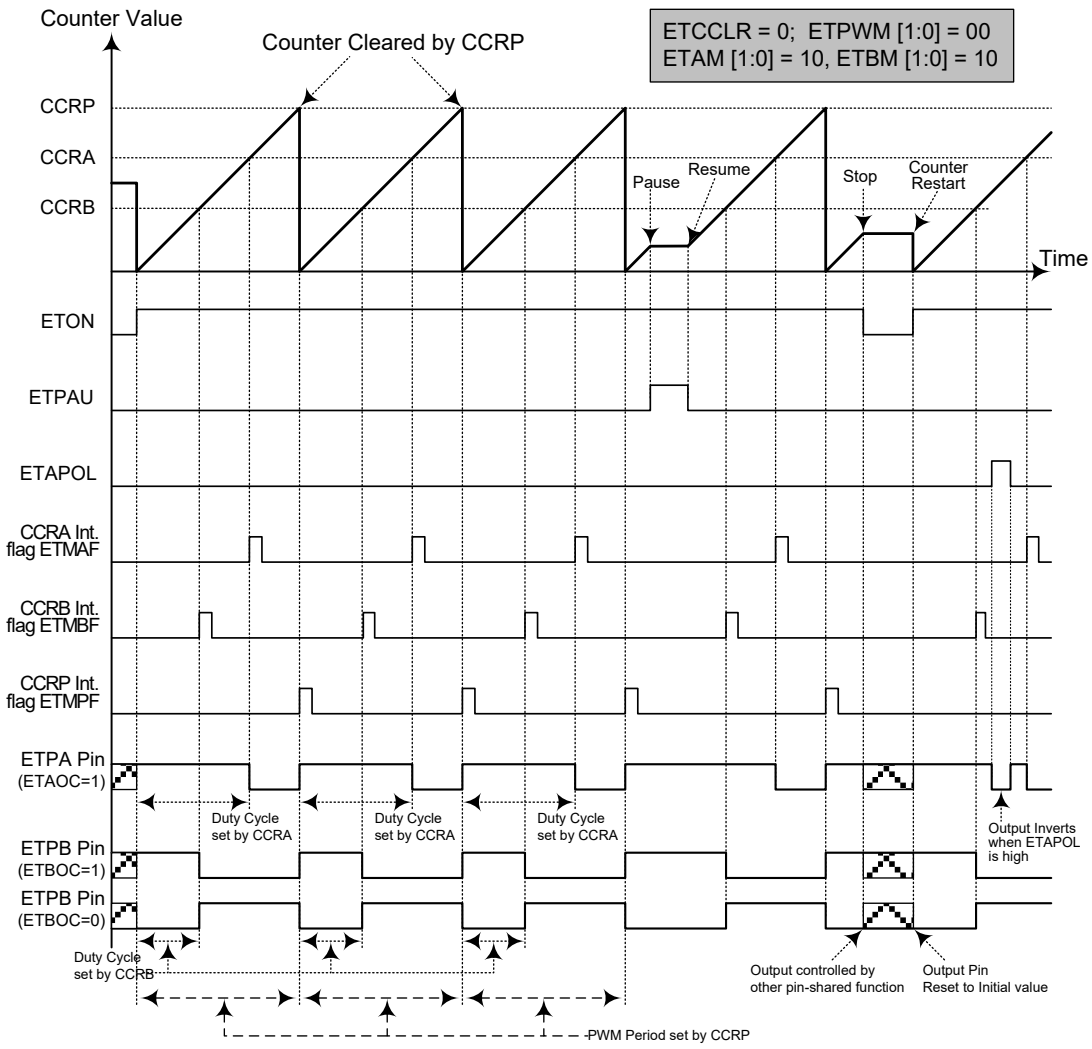
CCRA	1	2	3	---	511	512	---	---	1021	1022	1023
Period	1	2	3	---	511	512	---	---	1021	1022	1024
B Duty	CCRB										

- **10-bit ETM, PWM Mode, Centre-aligned Mode, ETCCLR=0**

CCRP	001b	011b	011b	100b	101b	110b	111b	000b
Period	256	512	768	1024	1280	1536	1792	2046
A Duty	(CCRA×2)-1							
B Duty	(CCRB×2)-1							

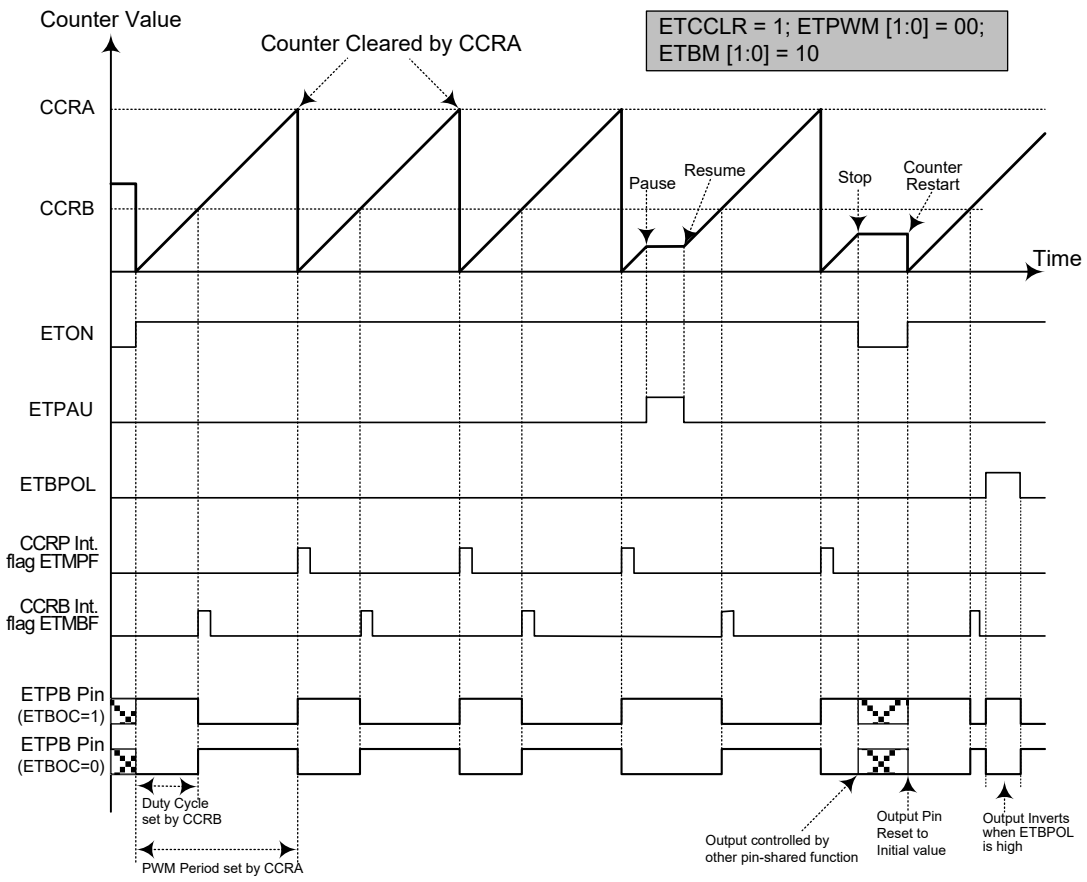
- **10-bit ETM, PWM Mode, Edge-aligned Mode, ETCCLR=1**

CCRA	1	2	3	---	511	512	---	---	1021	1022	1023
Period	2	4	6	---	1022	1024	---	---	2042	2044	2046
B Duty	(CCRB×2)-1										



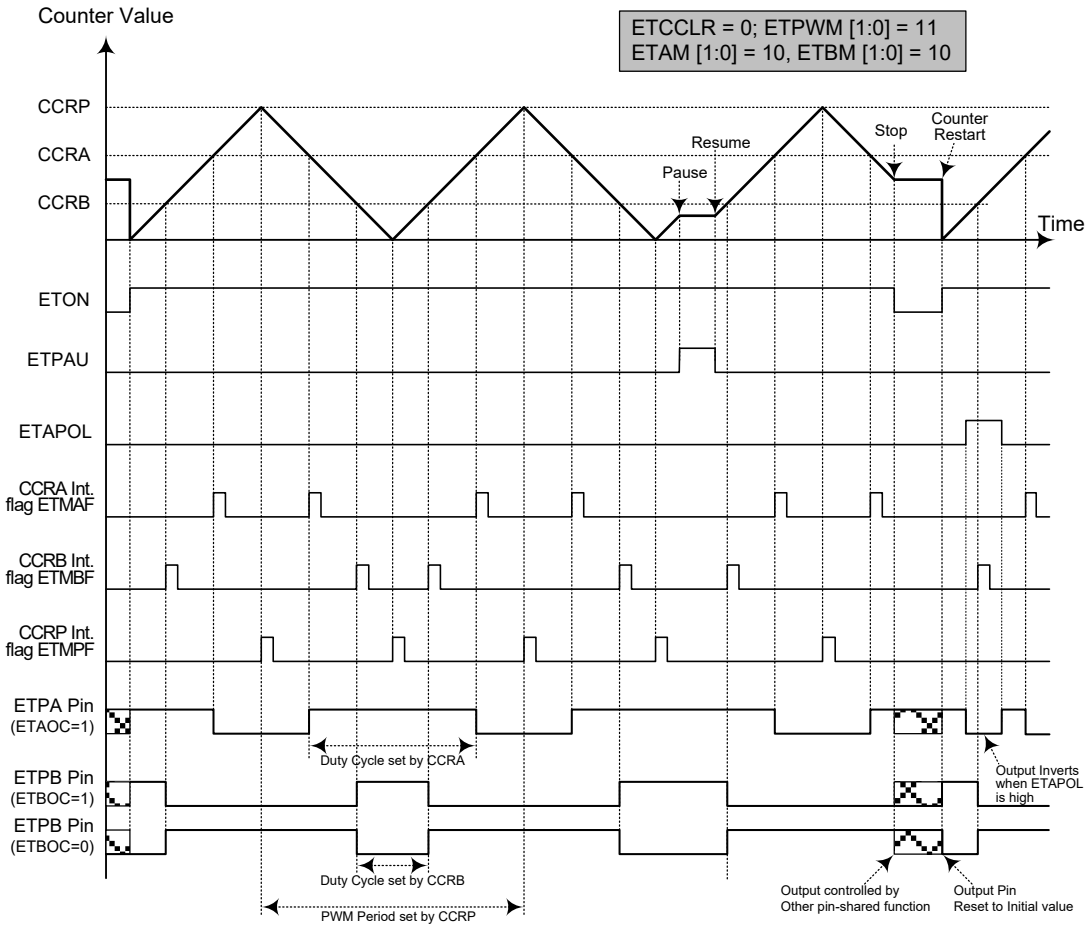
10-bit ETM PWM Mode – Edge Aligned

Note: 1. Here ETCCLR=0 therefore CCRP clears counter and determines the PWM period
2. The internal PWM function continues running even when ETAIO [1:0] (or ETBIO [1:0]) = 00 or 01
3. CCRA controls the ETPA PWM duty and CCRB controls the ETPB PWM duty



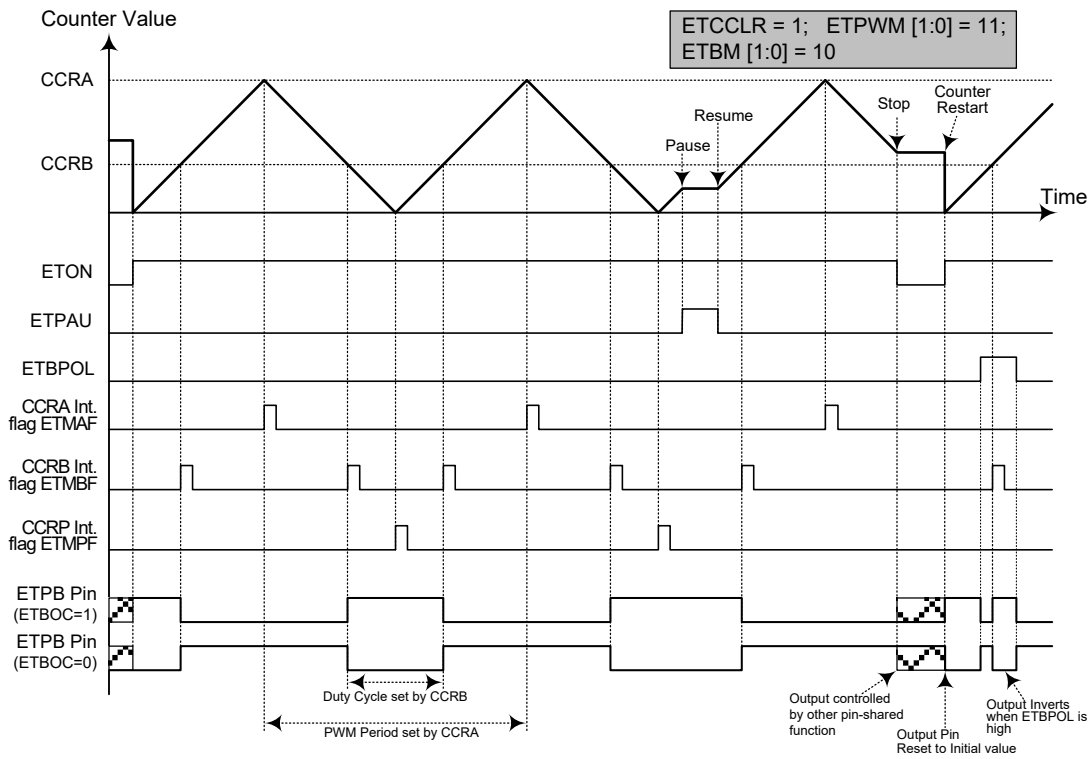
10-bit ETM PWM Mode – Edge Aligned

Note: 1. Here ETCCLR=1 therefore CCRA clears counter and determines the PWM period
2. The internal PWM function continues running even when ETBIO [1:0] = 00 or 01
3. The CCRA controls the ETPB PWM period and CCRB controls the ETPB PWM duty
4. Here the ETPA pin control register should not enable the ETPA pin as an ETM output pin



10-bit ETM PWM Mode – Centre Aligned

Note: 1. Here ETCCLR=0 therefore CCRP clears counter and determines the PWM period
2. ETPWM [1:0] = 11 therefore the PWM is centre aligned
3. The internal PWM function continues running even when ETAIO [1:0] (or ETBIO [1:0]) = 00 or 01
4. The CCRA controls the ETPA PWM duty and CCRB controls the ETPB PWM duty
5. CCRP will generate an interrupt request when the counter decrements to its zero value



10-bit ETM PWM Mode – Centre Aligned

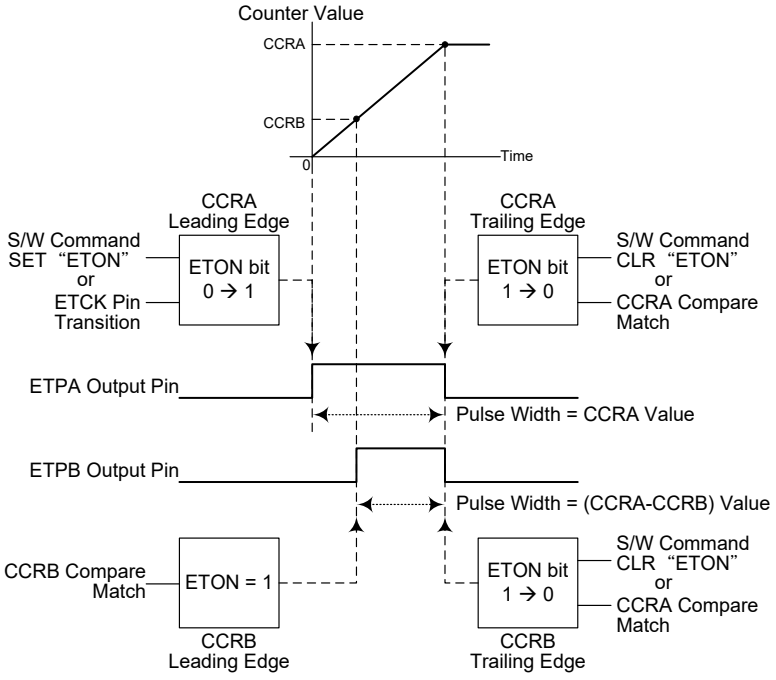
Note: 1. Here ETCCLR=1 therefore CCRA clears counter and determines the PWM period
2. ETPWM [1:0] = 11 therefore the PWM is centre aligned
3. The internal PWM function continues running even when ETBIO [1:0] = 00 or 01
4. The CCRA controls the ETPB PWM period and CCRB controls the ETPB PWM duty
5. CCRP will generate an interrupt request when the counter decrements to its zero value

Single Pulse Output Mode

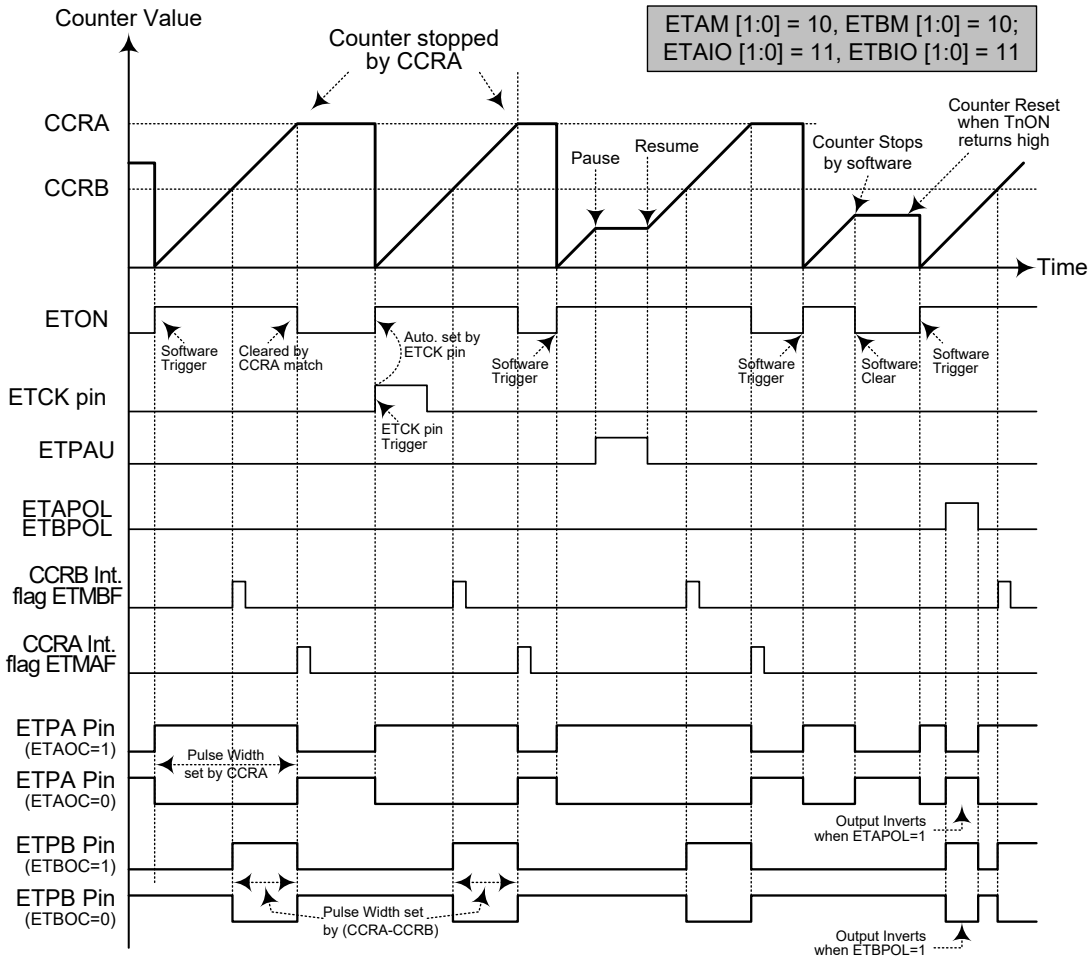
To select this mode, the required bit pairs, ETAM1, ETAM0 and ETBM1, ETBM0 should be set to 10 respectively and also the corresponding ETAIO1, ETAIO0 and ETBIO1, ETBIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the ETM output pin.

The trigger for the pulse ETPA output leading edge is a low to high transition of the ETON bit, which can be implemented using the application program. The trigger for the pulse ETPB output leading edge is a compare match from Comparator B, which can be implemented using the application program. However in the Single Pulse Mode, the ETON bit can also be made to automatically change from low to high using the external ETCK pin, which will in turn initiate the Single Pulse output of ETPA. When the ETON bit transitions to a high level, the counter will start running and the pulse leading edge of ETPA will be generated. The ETON bit should remain high when the pulse is in its active state. The generated pulse trailing edge of ETPA and ETPB will be generated when the ETON 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 ETON bit and thus generate the Single Pulse output trailing edge of ETPA and ETPB. In this way the CCRA value can be used to control the pulse width of ETPA. The (CCRA-CCRB) value can be used to control the pulse width of ETPB. A compare match from Comparator A and Comparator B will also generate ETM interrupts. The counter can only be reset back to zero when the ETON bit changes from low to high when the counter restarts. In the Single Pulse Mode CCRP is not used. The ETCCLR bit is also not used.



Single Pulse Generation



10-bit ETM Single Pulse Output Mode

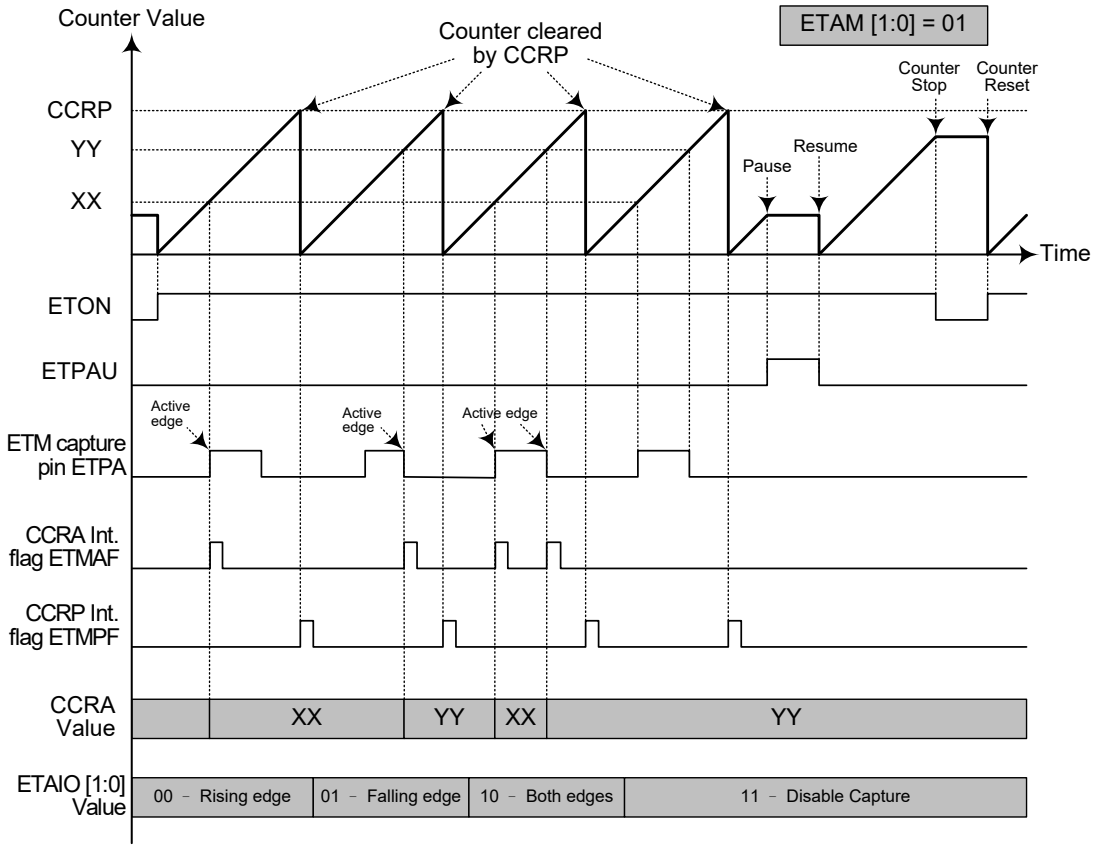
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse triggered by the ETCK pin or by setting the ETON bit high
4. An ETCK pin active edge will automatically set the ETON bit high.
5. In the Single Pulse Mode, ETAIO [1:0] and ETBIO [1:0] must be set to “11” and can not be changed.

Capture Input Mode

To select this mode bits ETAM1, ETAM0 and ETBM1, ETBM0 in the ETMC1 and ETMC2 registers 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 ETPIA and ETPIB pins, 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 ETAIO1, ETAIO0 and ETBIO1, ETBIO0 bits in the ETMC1 and ETMC2 registers. The counter is started when the ETON bit changes from low to high which is initiated using the application program.

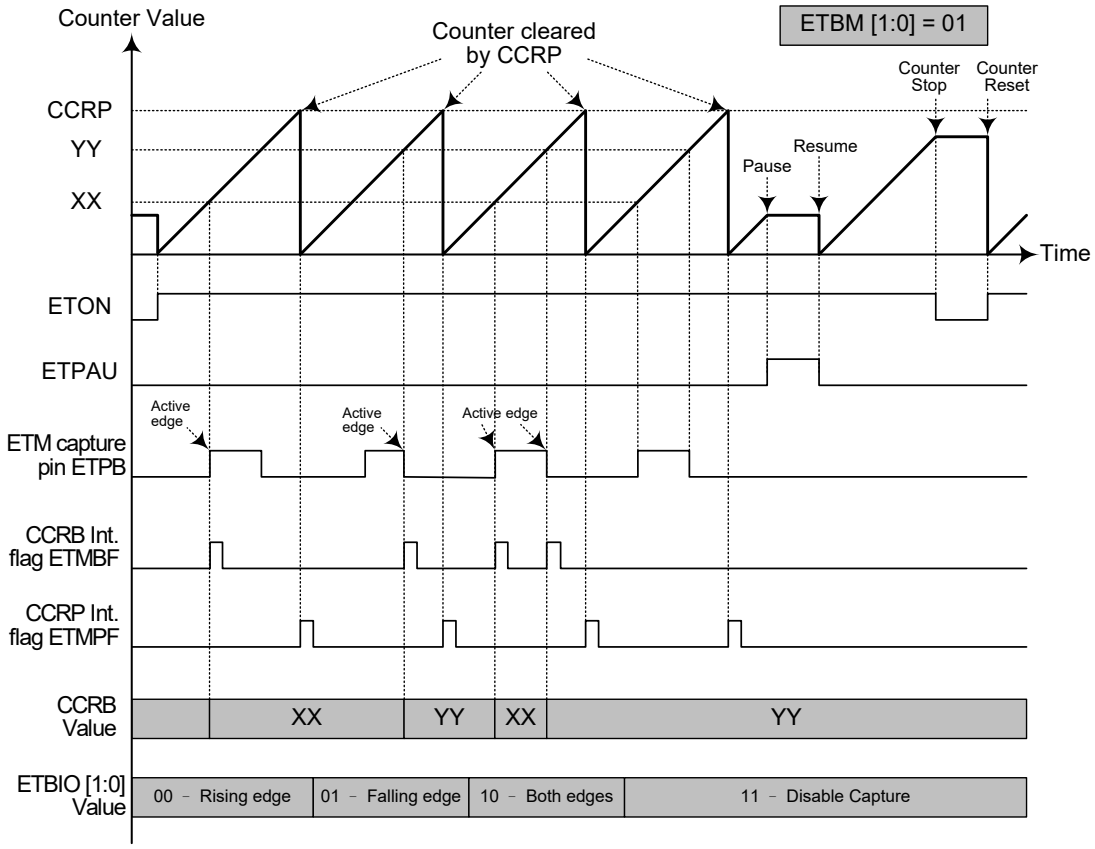
When the required edge transition appears on the ETPIA and ETPIB pins, the present value in the counter will be latched into the CCRA and CCRB registers and an ETM interrupt generated. Irrespective of what events occur on the ETPIA and ETPIB pins the counter will continue to free run until the ETON bit 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, an ETM 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 ETAIO1, ETAIO0 and ETBIO1, ETBIO0 bits can select the active trigger edge on the ETPIA and ETPIB pins to be a rising edge, falling edge or both edge types. If the ETAIO1, ETAIO0 and ETBIO1, ETBIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the ETPIA and ETPIB pins, however it must be noted that the counter will continue to run.

As the ETPIA and ETPIB pins are pin shared with other functions, care must be taken if the ETM 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 ETCCLR, ETAOC, ETBOC, ETAPOL and ETBPOL bits are not used in this mode.



10-bit ETM CCRA Capture Input Mode

Note: 1. ETAM [1:0] = 01 and active edge set by the ETAIO [1:0] bits
2. An ETM Capture input pin active edge transfers the counter value to CCRA
3. ETCCLR bit not used
4. No output function – ETAOC and ETAPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero.



10-bit ETM CCRB Capture Input Mode

Note: 1. ETBM [1:0] = 01 and active edge set by the ETBIO [1:0] bits
2. An ETM Capture input pin active edge transfers the counter value to CCRA
3. ETCCLR bit not used
4. No output function – ETBOC and ETBPOL 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

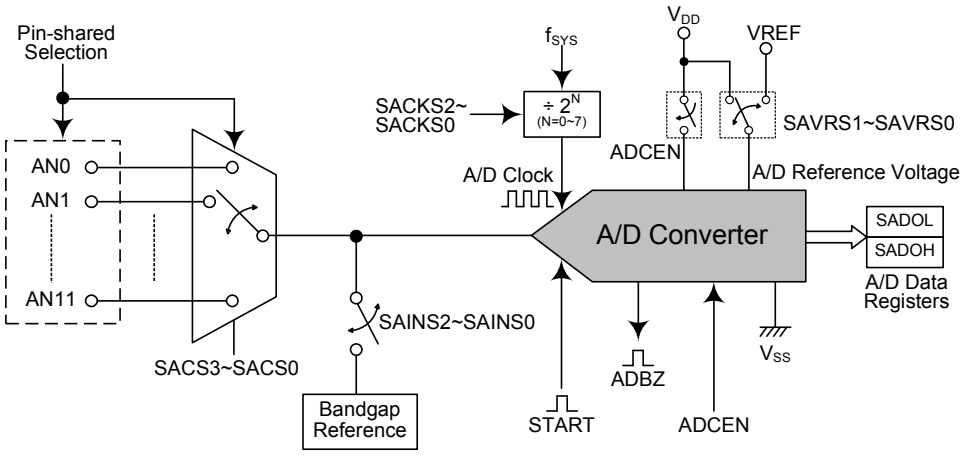
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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 Overview

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the Bandgap reference voltage, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 bits together with the SACS3~SACS0 bits. Note that when the internal analog signal is to be converted, the pin-shared control bits should also be properly configured except the SAINS and SACS bit fields. More detailed information about the A/D input signal is described in the "A/D Converter Control Registers" and "A/D Converter Input Signal" sections respectively.

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

Device	Input Channel	A/D Channel Select Bits	Input Pins
HT67F60A/HT67F70A	12	SACS3~SACS0	AN0~AN11



A/D Converter Structure

A/D Converter Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS10	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

A/D Converter Registers List

A/D Converter Data Registers – SADOL, SADOH

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

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

A/D Converter Data Registers

A/D Converter Control Registers – SADC0, SADC1

To control the function and operation of the A/D converter, two control registers known as SADC0, SADC1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input. If the SAINS2~SAINS0 bits are set to “000”, the external analog channel input is selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected to be converted. If the SAINS2~SAINS0 bits are set to “001”, the internal Bandgap reference voltage is selected to be converted. Care must be taken when the internal analog signal is selected to be converted. If the internal analog signal is selected to be converted, the SACS3~SACS0 bits must be set to a value of “11xx”. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

SAINS [2:0]	SACS [3:0]	Input Signals	Description
000, 101, 110, 111	0000~1011	AN0~AN11	External pin analog input
	11xx	—	Floating
001	11xx	V_{BG}	Internal Bandgap reference voltage
010~100	xxxx	—	Reserved

A/D Converter Input Signal Selection

- **SADC0 Register**

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

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

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

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

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

Bit 3~0 **SACS3~SACS0**: A/D converter external analog input channel select
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000: AN8
1001: AN9
1010: AN10
1011: AN11
11xx: Floating

- **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS10	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~5 **SAINS2~SAINS0**: A/D converter input signal select
000: External source – External analog channel input
001: Internal source – Internal Bandgap reference voltage
010~100: Reserved
101~111: External source – External analog channel input
Care must be taken if the SAINS2~SAINS0 bits are set to "001" to select the internal analog signal to be converted. When the internal analog signal is selected to be converted, the SACS3~SACS0 bits must be set to a value of "11xx". Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage select
00: From VREF pin
01: From VDD pin
1x: From VREF pin

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source select
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$
These bits are used to select the clock source for the A/D converter.

A/D Operation

The START bit is used to start the A/D conversion. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the A/D conversion will not be initiated.

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in progress or not. This bit will be automatically set to 1 by the microcontroller after an A/D conversion is successfully initiated. When the A/D conversion is complete, the ADBZ will be cleared to 0. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can poll the ADBZ bit in the SADC0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

Although the A/D clock source is determined by the system clock f_{SYS}, and by bits SACKS2~SADCKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken for selected system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the SACKS2~SADCKS0 bits should not be set to 000 or 11x. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values.

f_{SYS}	A/D Clock Period (t_{ADCK})							
	ADCK[2:0] = 000 (f_{SYS})	ADCK[2:0] = 001 (f_{SYS}/2)	ADCK[2:0] = 010 (f_{SYS}/4)	ADCK[2:0] = 011 (f_{SYS}/8)	ADCK[2:0] = 100 (f_{SYS}/16)	ADCK[2:0] = 101 (f_{SYS}/62)	ADCK[2:0] = 110 (f_{SYS}/64)	ADCK[2:0] = 111 (f_{SYS}/128)
1 MHz	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *	128μs *
2 MHz	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *
4 MHz	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *
8 MHz	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *
12 MHz	83ns *	167ns *	333ns *	667ns	1.33μs	2.67μs	5.33μs	10.67μs *
16 MHz	62.5ns *	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs
20 MHz	50ns *	100ns *	200ns *	400ns *	800ns	1.6μs	3.2μs	6.4μs

A/D Clock Period Examples

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

The reference voltage supply to the A/D Converter can be supplied from either the positive power supply pin, VDD, or from an external reference sources supplied on pin VREF. The desired selection is made using the SAVRS1 and SAVRS0 bits. When the SAVRS bit field is set to "01", the A/D converter reference voltage will come from the VDD pin. Otherwise, if the SAVRS bit field is set to any other value except "01", the A/D converter reference voltage will come from the VREF pin. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bits should be properly configured to disable other pin functions.

SAVRS [1:0]	Reference Voltage	Description
00	V_{REF}	A/D Converter Reference voltage comes from VREF pin
01	V_{DD}	A/D Converter Reference voltage comes from VDD pin
1x	V_{REF}	A/D Converter Reference voltage comes from VREF pin

A/D Converter Reference Voltage Selection

A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins as well as other functions. The corresponding pin-shared function selection bits for each pin in the PxS1 and PxS0 registers, determine whether the external pins are setup as A/D converter analog channel inputs or they have other functions. If the corresponding pin is setup to be an A/D converter analog channel input, the original pin functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the relevant pin-shared function selection bits enable an A/D analog channel input, the status of the port control register will be overridden.

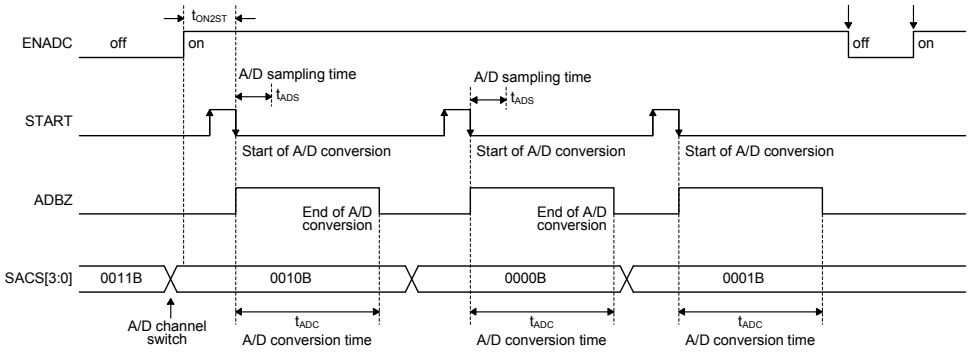
The A/D converter has its own reference voltage pin, VREF. However, the reference voltage can also be supplied from the power supply pin, a choice which is made through the SAVRS1 and SAVRS0 bits in the SADC1 register. The analog input values must not be allowed to exceed the value of V_{REF}.

Conversion Rate and Timing Diagram

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

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

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



A/D Conversion Timing

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by properly programming the SACKS2~SACKS0 bits in the SADC1 register.
- Step 2
  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to one.
- Step 3
  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS2~SAINS0 bits.
  Select the external channel input to be converted, go to Step 4.
  Select the internal analog signal to be converted, go to Step 5.
- Step 4
  If the A/D input signal comes from the external channel input selecting by configuring the SAINS bit field, the corresponding pins should first be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS bit field. After this step, go to Step 6.

- Step 5
  Before the A/D input signal is selected to come from the internal analog signal by configuring the SAINS bit field, the SACS bit field must be first configured to a value of "11xx" to disconnect the external channel input. The desired internal analog signal then can be selected by configuring the SAINS bit field. After this step, go to Step 6.
- Step 6
  Select the reference voltgage source by configuring the SAVRS1~SAVRS0 bits.
- Step 7
  Select the A/D converter output data format by configuring the ADRFS bit.
- Step 8
  If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt bontrol bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 9
  The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10
  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.
  Note: When checking for the end of the conversion process, if the method of polling the ADBZ bit in the SADC0 register is used, the interrupt enable step above can be omitted.

Programming Considerations

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

A/D Transfer Function

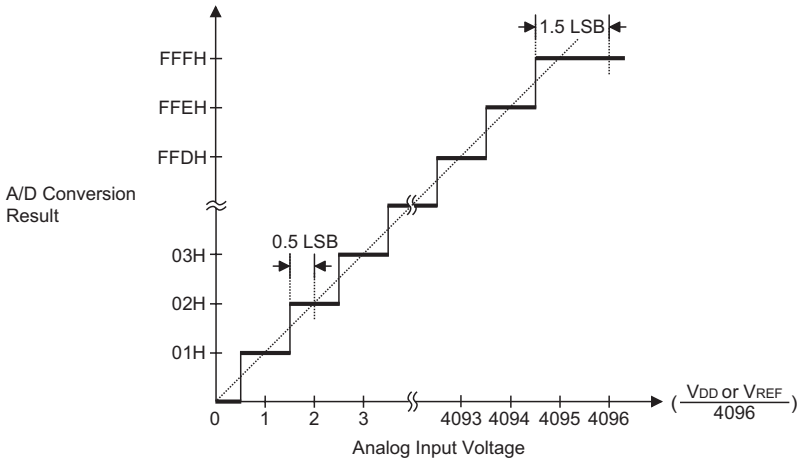
As the devices contain a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the V_{DD} or V_{REF} voltage, this gives a single bit analog input value of V_{DD} or V_{REF} divided by 4096.

$$1\ \text{LSB} = (V_{DD} \text{ or } V_{REF}) \div 4096$$

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

$$\text{A/D input voltage} = \text{A/D output digital value} \times (V_{DD} \text{ or } V_{REF}) \div 4096$$

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the V_{DD} or V_{REF} level.



Ideal A/D Transfer Function

A/D Programming Examples

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

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

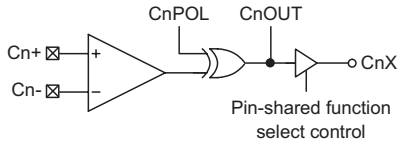
```
clr  ADE              ; disable ADC interrupt
mov  a,03H
mov  SADC1,a          ; select fSYS/8 as A/D clock
set  ADCEN
mov  a,0CH            ; setup PAS0 to configure pin AN0
mov  PAS0,a
mov  a,20H
mov  SADC0,a          ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr  START            ; high pulse on start bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
:
polling_EOC:
sz   ADBZ             ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp  polling_EOC      ; continue polling
:
mov  a,SADOL          ; read low byte conversion result value
mov  ADRL_buffer,a    ; save result to user defined register
mov  a,SADOH          ; read high byte conversion result value
mov  ADRH_buffer,a    ; save result to user defined register
:
jmp  start_conversion ; start next A/D conversion
```

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

```
clr  ADE              ; disable ADC interrupt
mov  a,03H
mov  SADC1,a          ; select fSYS/8 as A/D clock
set  ADCEN
mov  a,0Ch            ; setup PAS0 to configure pin AN0
mov  PAS0,a
mov  a,20h
mov  SADC0,a          ; enable and connect AN0 channel to A/D converter
:
Start_conversion:
clr  START            ; high pulse on START bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
clr  ADF              ; clear ADC interrupt request flag
set  ADE              ; enable ADC interrupt
set  EMI              ; enable global interrupt
:
:
ADC_ISR:              ; ADC interrupt service routine
mov  acc_stack,a      ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a   ; save STATUS to user defined memory
:
mov  a, SADOL         ; read low byte conversion result value
mov  adrl_buffer,a    ; save result to user defined register
mov  a, SADOH         ; read high byte conversion result value
mov  adrh_buffer,a    ; save result to user defined register
:
EXIT_INT_ISR:
mov  a,status_stack
mov  STATUS,a         ; restore STATUS from user defined memory
mov  a,acc_stack      ; restore ACC from user defined memory
reti
```

Comparators

Two independent analog comparators are contained within these devices. These functions offer flexibility via their register controlled features such as power-down, polarity select, hysteresis etc. In sharing their pins with normal I/O pins the comparators do not waste precious I/O pins if there functions are otherwise unused.



Comparator

Comparator Operation

The device contains two comparator functions which are used to compare two analog voltages and provide an output based on their difference. Full control over the two internal comparators is provided via two control registers, CP0C and CP1C, one assigned to each comparator. The comparator output is recorded via a bit in their respective control register, but can also be transferred out onto a shared I/O pin. Additional comparator functions include, output polarity, hysteresis functions and power down control.

Any pull-high resistors connected to the shared comparator input pins will be automatically disconnected when the comparator is enabled. As the comparator inputs approach their switching level, some spurious output signals may be generated on the comparator output due to the slow rising or falling nature of the input signals. This can be minimised by selecting the hysteresis function will apply a small amount of positive feedback to the comparator. Ideally the comparator should switch at the point where the positive and negative inputs signals are at the same voltage level, however, unavoidable input offsets introduce some uncertainties here. The hysteresis function, if enabled, also increases the switching offset value.

Comparator Registers

There are two registers for overall comparator operation, one for each comparator. As corresponding bits in the two registers have identical functions, the following table applies to both registers.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CP0C	—	C0EN	C0POL	C0OUT	—	—	—	C0HYEN
CP1C	—	C1EN	C1POL	C1OUT	—	—	—	C1HYEN

Comparator Registers List

CP0C Register

Bit	7	6	5	4	3	2	1	0
Name	—	C0EN	C0POL	C0OUT	—	—	—	C0HYEN
R/W	—	R/W	R/W	R	—	—	—	R/W
POR	—	0	0	0	—	—	—	1

Bit 7 Unimplemented, read as "0"

Bit 6 **C0EN**: Comparator On/Off control
0: Off
1: On
This is the Comparator On/Off control. If the bit is zero, the comparator will be switched off and no power consumed even if analog voltages are applied to its inputs. For power sensitive applications this bit should be cleared to zero if the comparator is not used or before the device enters the SLEEP or IDLE mode.

Bit 5 **C0POL**: Comparator output polarity control
0: Not inverted
1: Inverted
This is the Comparator polarity bit. If the bit is zero, then the C0OUT bit will reflect the non-inverted output condition of the comparator. If the bit is high, the comparator C0OUT bit will be inverted.

Bit 4 **C0OUT**: Comparator output
C0POL=0
0: C0+ < C0–
1: C0+ > C0–
C0POL=1
0: C0+ > C0–
1: C0+ < C0–
This bit stores the comparator output bit. The polarity of the bit is determined by the voltages on the comparator inputs and by the condition of the C0POL bit.

Bit 3~1 Unimplemented, read as "0"

Bit 0 **C0HYEN**: Comparator Hysteresis control
0: Off
1: On
This is the hysteresis control bit and if set high will apply a limited amount of hysteresis to the comparator, as specified in the Comparator Electrical Characteristics table. The positive feedback induced by hesteresus reduces the effect of spurious switching near the comparator threshold.

CP1C Register

Bit	7	6	5	4	3	2	1	0
Name	—	C1EN	C1POL	C1OUT	—	—	—	C1HYEN
R/W	—	R/W	R/W	R	—	—	—	R/W
POR	—	0	0	0	—	—	—	1

Bit 7 Unimplemented, read as "0"

Bit 6 **C1EN**: Comparator On/Off control
0: Off
1: On
This is the Comparator On/Off control. If the bit is zero, the comparator will be switched off and no power consumed even if analog voltages are applied to its inputs. For power sensitive applications this bit should be cleared to zero if the comparator is not used or before the device enters the SLEEP or IDLE mode.

Bit 5 **C1POL**: Comparator output polarity control
0: Not inverted
1: Inverted
This is the Comparator polarity bit. If the bit is zero, then the C1OUT bit will reflect the non-inverted output condition of the comparator. If the bit is high, the comparator C1OUT bit will be inverted.

Bit 4 **C1OUT**: Comparator output
C1POL=0
0: C1+ < C1–
1: C1+ > C1–
C1POL=1
0: C1+ > C1–
1: C1+ < C1–
This bit stores the comparator output bit. The polarity of the bit is determined by the voltages on the comparator inputs and by the condition of the C1POL bit.

Bit 3~1 Unimplemented, read as "0"

Bit 0 **C1HYEN**: Comparator Hysteresis control
0: Off
1: On
This is the hysteresis control bit and if set high will apply a limited amount of hysteresis to the comparator, as specified in the Comparator Electrical Characteristics table. The positive feedback induced by hesteresus reduces the effect of spurious switching near the comparator threshold.

Comparator interrupt

Each also possesses its own interrupt function. When any one of the changes state, its relevant interrupt flag will be set, and if the corresponding interrupt enable bit is set, then a jump to its relevant interrupt vector will be executed. Note that it is the changing state of the C0OUT or C1OUT bit and not the output pin which generates an interrupt. If the microcontroller is in the SLEEP or IDLE Mode and the Comparator is enabled, then if the external input lines cause the Comparator output to change state, the resulting generated interrupt flag will also generate a wake-up. If it is required to disable a wake-up from occurring, then the interrupt flag should be first set high before entering the SLEEP or IDLE Mode.

Programming Considerations

If the comparator is enabled, it will remain active when the microcontroller enters the SLEEP or IDLE Mode. However, as it will consume a certain amount of power, the user may wish to consider disabling it before the SLEEP or IDLE Mode is entered.

As comparator pins are shared with normal I/O pins the I/O registers for these pins will be read as zero (port control register is "1") or read as port data register value (port control register is "0") if the comparator function is enabled.

Serial Interface Module – SIM

The device contains a Serial Interface Module, which includes both the four-line SPI interface and two-line I²C 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 or I²C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins and therefore the SIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As both interface types share the same pins and registers, the choice of whether the SPI or I²C type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O pins are selected using pull-high control registers when the SIM function is enabled and the corresponding pins are used as SIM 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 devices can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, these devices provided 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.

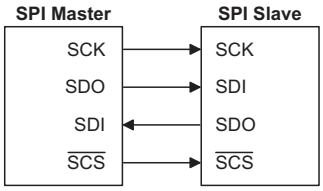
SPI Interface Operation

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, SCK 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 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. After the desired SPI configuration has been set it can be disabled or enabled using the SIMEN bit in the SIMC0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the 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.

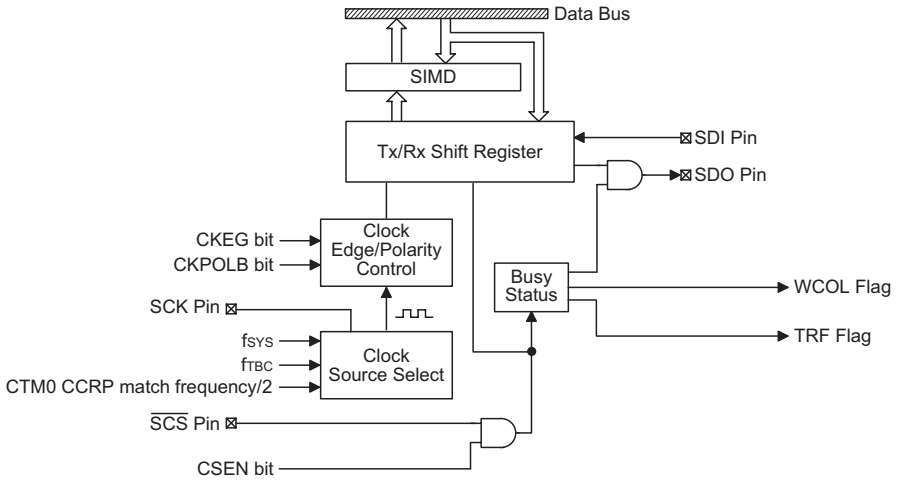
The SPI function in this device offers the following features:

- 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 Master/Slave Connection



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 registers SIMC0 and SIMC2. Note that the SIMC1 register is only used by the I²C interface.

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

SPI Registers List

SIMD 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.

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

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I²C function. The SIMC1 register is not used by the SPI function, only by the I²C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Register SIMC2 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	—	SIMDEB1	SIMDEB0	SIMEN	—
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	—
POR	1	1	1	—	0	0	0	—

Bit 7~5 **SIM2~SIM0**: SIM 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_{TBC}
100: SPI master mode; SPI clock is CTM0 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Non SIM function
These bits setup the overall operating mode of the SIM 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 CTM0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

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

Bit 1 **SIMEN**: SIM Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective.If the SIM is configured to operate as an SPI interface via the 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 SIM is configured to operate as an I²C interface via the 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 Unimplemented, read as "0"

- **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 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 $\overline{SCS}$ pin control

0: Disable

1: Enable

The CSEN bit is used as an enable/disable for the $\overline{SCS}$ pin. If this bit is low, then the $\overline{SCS}$ pin will be disabled and placed into I/O pin or other pin-shared functions. If the bit is high, the $\overline{SCS}$ 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 whether a data collision has occurred or not. If this bit is high, it means that data has been attempted to be written to the SIMD register duting a data transfer operation. This writing operation will be ignored if data is being transferred. This bit can be cleared by the application program.

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

0: SPI data is being transferred

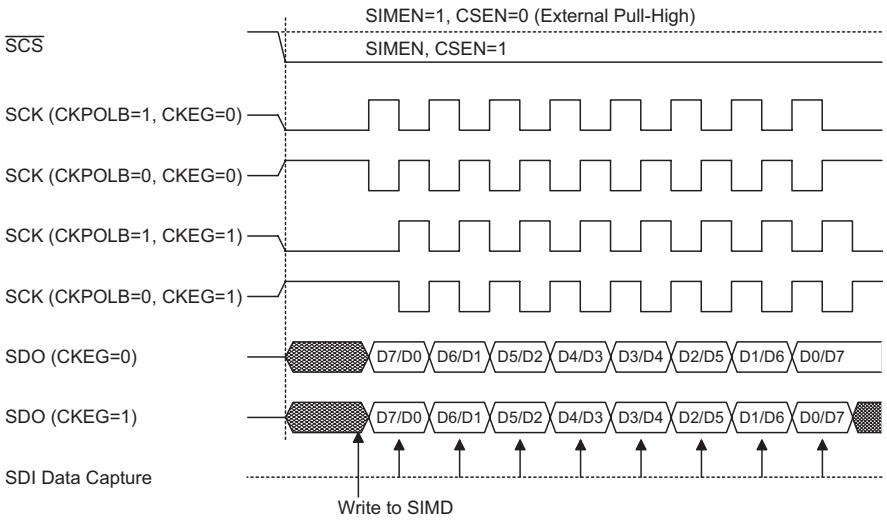
1: SPI data transfer is completed

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

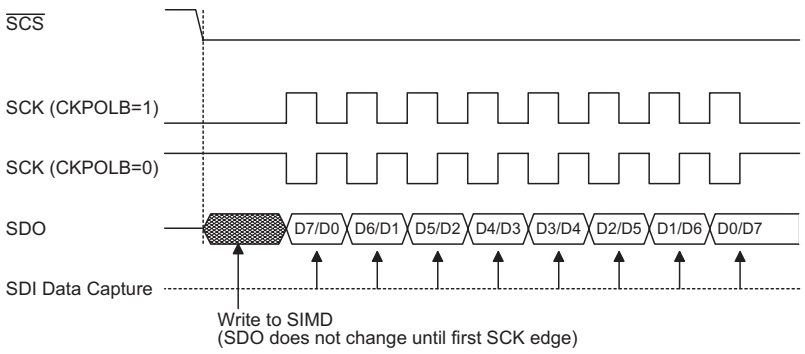
SPI Communication

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 complete, the TRF 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 SIMD register will be transmitted and any data on the SDI pin will be shifted into the SIMD register. The master should output a $\overline{SCS}$ 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 $\overline{SCS}$ signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagram shows the relationship between the slave data and $\overline{SCS}$ signal for various configurations of the CKPOLB and CKEG bits.

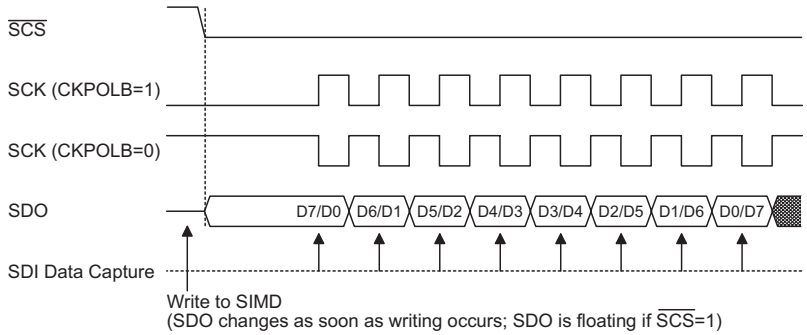
The SPI will continue to function even in the IDLE Mode.



SPI Master Mode Timing

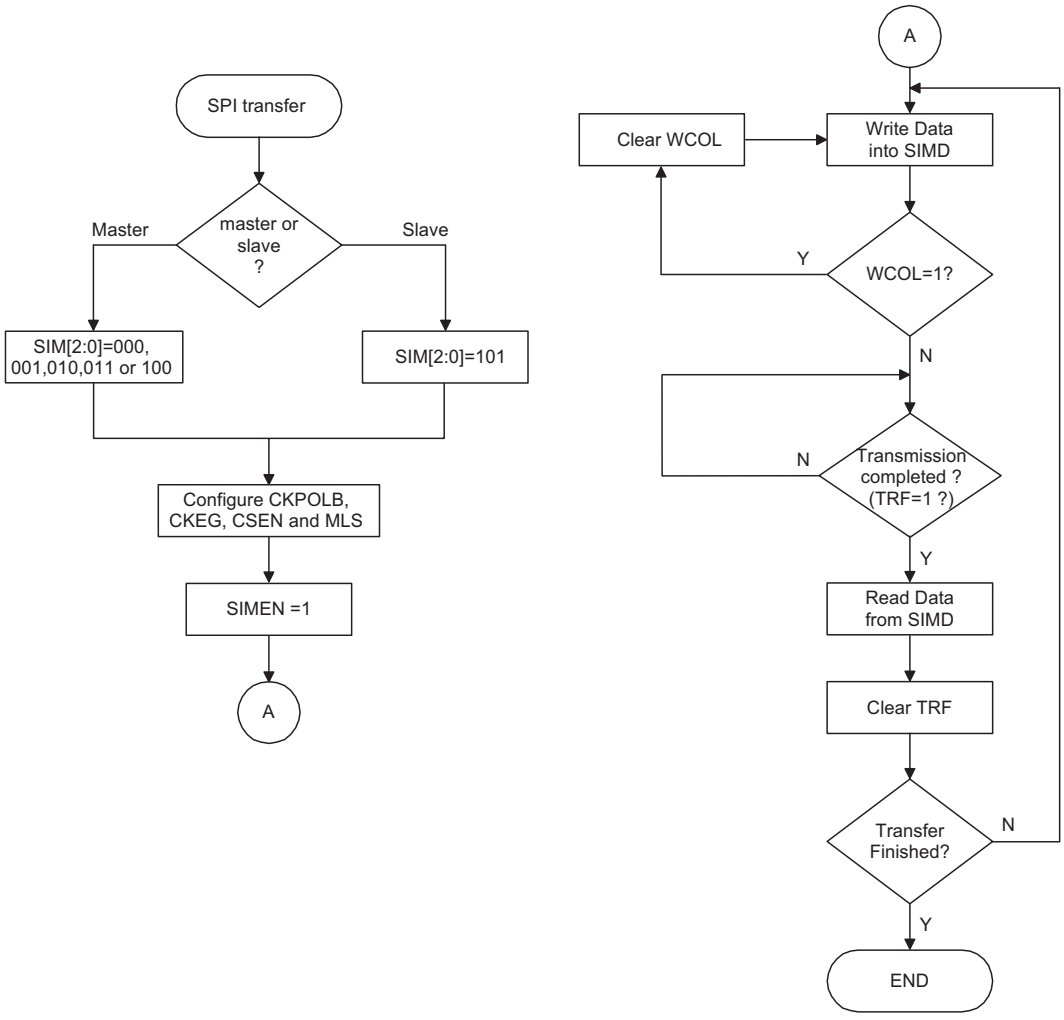


SPI Slave Mode Timing – CKEG = 0



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

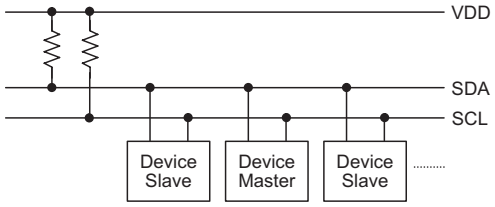
SPI Slave Mode Timing – CKEG = 1



SPI Transfer Control Flow Chart

I²C Interface

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

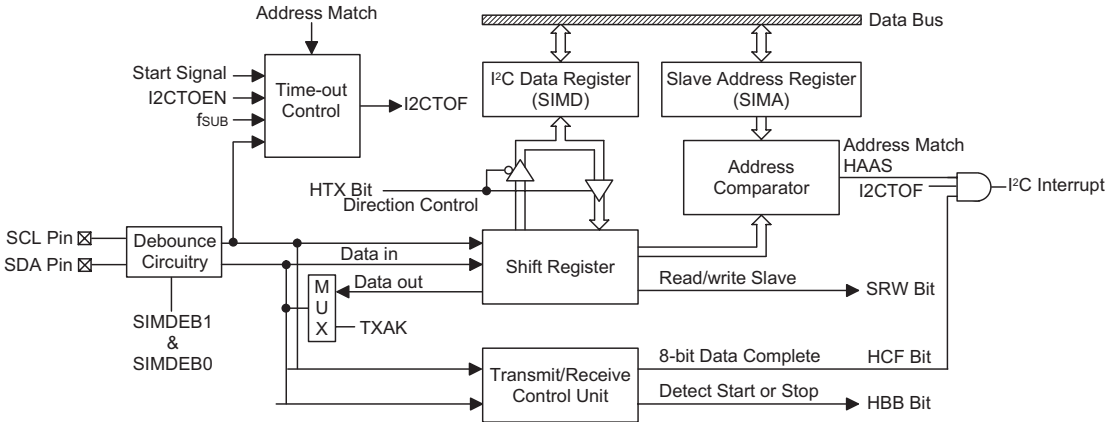


I²C Master Slave Bus Connection

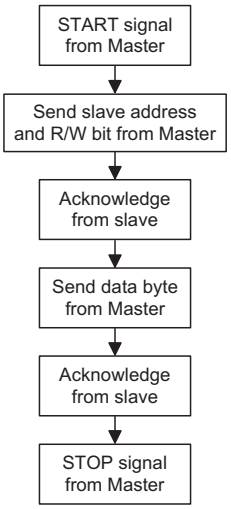
I²C interface Operation

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

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For these devices, which only operate in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode.



I²C Block Diagram



The SIMDBC1 and SIMDBC0 bits determine the debounce time of the I²C interface. This uses the system 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 Devounce	f_{SYS} > 2 MHz	f_{SYS} > 5 MHz
2 system clock debounce	f_{SYS} > 4 MHz	f_{SYS} > 10 MHz
4 system clock debounce	f_{SYS} > 8 MHz	f_{SYS} > 20 MHz

I²C Minimum f_{SYS} Frequency

I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMA, and one data register, SIMD. The SIMD register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I²C bus. Before the microcontroller 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 microcontroller can read it from the SIMD register. Any transmission or reception of data from the I²C bus must be made via the SIMD register. Note that the SIMA register also has the name SIMC2 which is used by the SPI function. Bit SIMEN and bits SIM2~SIM0 in register SIMC0 are used by the I²C interface.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	—
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	D0

I²C Registers List

- **SIMD 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.

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

- **SIMA 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. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	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 7~1 **IICA6~IICA0**: I²C slave address
IICA6~IICA0 is the I²C slave address bit 6 ~ bit 0

Bit 0 Undefined bit
The bit can be read or written by the application program.

There are also two control registers for the I²C interface, SIMC0 and SIMC1. The register SIMC0 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.

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: SIM 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_{TBC}
100: SPI master mode; SPI clock is CTM0 CCRP match frequency/2
101: SPI slave mode
110: I^2C slave mode
111: Non SIM function
These bits setup the overall operating mode of the SIM function. As well as selecting if the I^2C 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 CTM0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

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

Bit 1 **SIMEN**: SIM Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I^2C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective.If the SIM is configured to operate as an SPI interface via the 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 SIM is configured to operate as an I^2C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I^2C 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 I^2C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 Unimplemented, read as "0"

- **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/W	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 data transfer completion 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 transmitter/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 does 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.

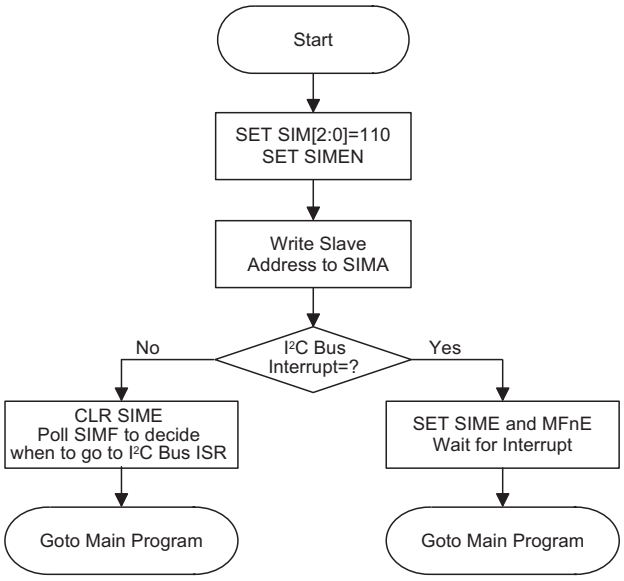
Bit 1 **IAMWU**: I²C Address Match Wake-Up control
0: Disable
1: Enable – must be cleared by the application program after wake-up
This bit should be set to 1 to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode 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 receives 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 I²C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS bit to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer. 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 SIM2~SIM0 and SIMEN bits in the SIMC0 register to "1" 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 SIME and SIM Muti-Function interrupt enable bit of the interrupt control register to enable the SIM interrupt and Multi-function 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.

Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the 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 I²C bus interrupt can come from two sources, when the program enters the interrupt subroutine, the HAAS bit 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. When a slave address is matched, the devices 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 slave 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.

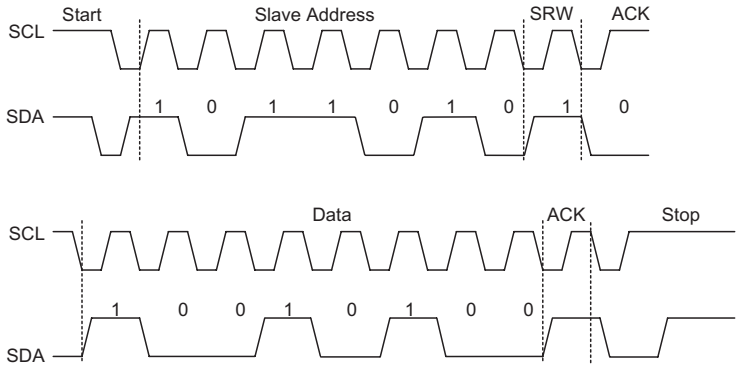
I²C Bus Slave Address Acknowledge Signal

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.

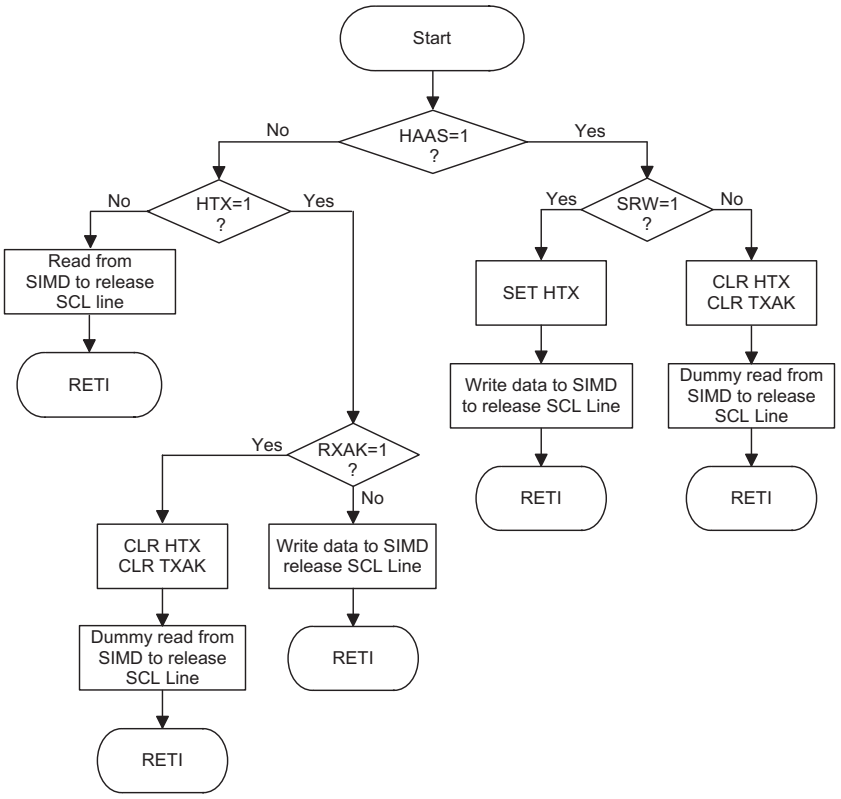


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	--------	S	SA	SR	M	D	A	D	A	--------	P

Note: * When a slave address is matched, the devices 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.

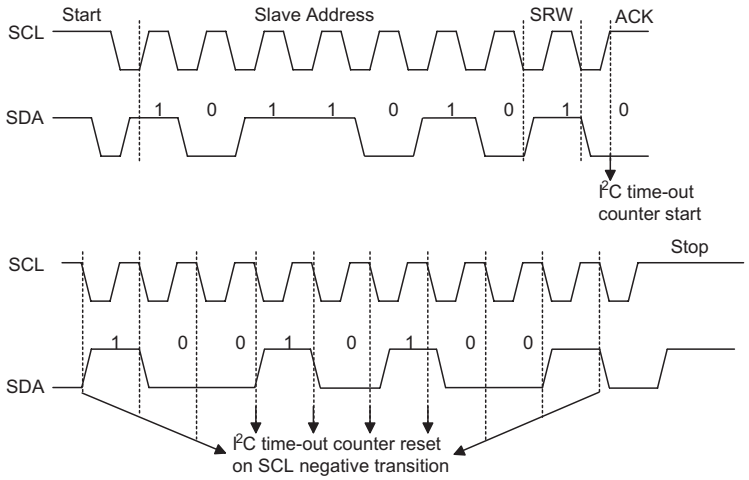
I²C Communication Timing Diagram



I²C Bus ISR Flow Chart

I²C Time-out Control

In order to reduce the I²C lockup problem due to reception of erroneous clock sources, a time-out function is provided. If the clock source connected to the I²C bus 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 to count 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 period specified by the I2CTOC 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 I2CTOEN bit will be cleared to zero and the I2CTF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the I²C interrrupt vector. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

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

I²C Register after Time-out

The I2CTOF flag can be cleared by the application program. There are 64 time-out period selections which can be selected using the I2CTOS bits in the I2CTOC register. The time-out duration is calculated by the formula: ((1~64) × (32/f_{SUB})). This gives a time-out period which ranges from about 1ms to 64ms.

- **I2CTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	I2CTOEN	I2CTOF	I2CTOS5	I2CTOS4	I2CTOS3	I2CTOS2	I2CTOS1	I2CTOS0
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 **I2CTOEN**: I²C Time-out control
0: Disable
1: Enable

Bit 6 **I2CTOF**: I²C Time-out flag
0: No time-out occurred
1: Time-out occurred

Bit 5~0 **I2CTOS5~I2CTOS0**: I²C Time-out period selection
I²C Time-out clock source is $f_{SUB}/32$

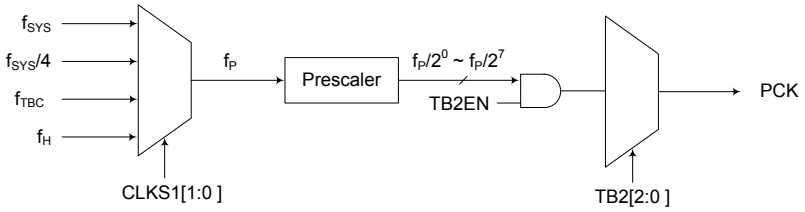
I²C Time-out period is equal to (I2CTOS[5:0]+1) × $\frac{32}{f_{SUB}}$

Peripheral Clock Output

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

Peripheral Clock Operation

As the peripheral clock output pin, PCK, is shared with I/O line, the required pin function is chosen using the relevant pin-shared function selection bit. The Peripheral Clock function is controlled using the TB2EN bit in the TBC2 register. The clock source for the Peripheral Clock Output can originate from the system clock f_{SYS}, the instruction clock, the high speed oscillator clock f_H or the f_{SUB} clock which can be selected by the CLKS11 and CLKS10 bits in the PSC1 register. The TB2EN bit in the TBC2 register is the overall on/off control, setting TB2EN bit to 1 enables the Peripheral Clock while setting TB2EN bit to 0 disables it. The required division ratio of the peripheral clock is selected using the TB22, TB21 and TB20 bits in the TBC2 register. If the peripheral clock source is switched off when the device enters the power down mode, this will disable the Peripheral Clock output.



Peripheral Clock Output

Peripheral Clock Registers

There are two internal registers which control the overall operation of the Peripheral Clock Output. These are the PSC1 and TBC2 registers.

Name	Bit							
	7	6	5	4	3	2	1	0
PSC1	—	—	—	—	—	—	CLKS11	CLKS10
TBC2	TB2EN	—	—	—	—	TB22	TB21	TB20

PCK Register List

PSC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKS11	CLKS10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKS11, CLKS10**: Peripheral Clock Source f_P selection
00: f_{SYS}
01: $f_{SYS}/4$
10: f_{TBC}
11: f_H

TBC2 Register

Bit	7	6	5	4	3	2	1	0
Name	TB2EN	—	—	—	—	TB22	TB21	TB20
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB2EN**: Peripheral Clock Function enable control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 1~0 **TB22, TB21, TB20**: Peripheral Clock output division selection
000: f_P
001: $f_P/2$
010: $f_P/4$
011: $f_P/8$
100: $f_P/16$
101: $f_P/32$
110: $f_P/64$
111: $f_P/128$

Serial Interface – SPIA

The device contains 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.

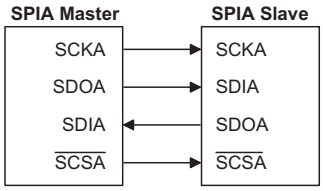
This SPIA interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices, etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying 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 SPIA interface specification can control multiple slave devices from a single master, this device is provided 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, SCKA is the Serial Clock line and $\overline{\text{SCSA}}$ is the Slave Select line. As the SPIA interface pins are pin-shared with other functions, the SPIA interface pins must first be selected by configuring the corresponding selection bits in the pin-shared function selection registers. The SPIA interface function is 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 utilised.

The $\overline{\text{SCSA}}$ pin is controlled by the application program, set the 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 an I/O function.

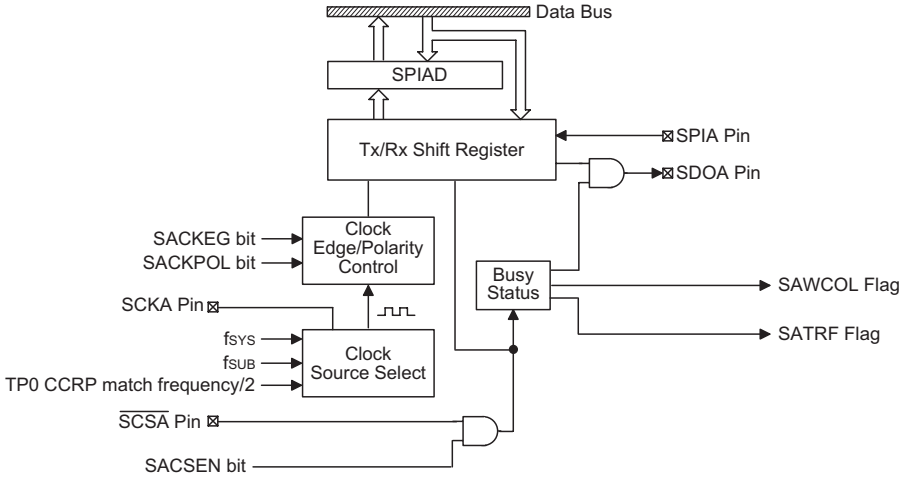


SPIA Master/Slave Connection

The SPIA Serial Interface function includes the following features:

- Full-duplex synchronous data transfer
- Both Master and Slave mode
- 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 SIMD data register and two registers SPIAC0 and SPIAC1.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SPIAC0	SASPI2	SASPIA1	SASPIA0	—	—	—	SPIAEN	—
SPIAC1	—	—	SACKPOLB	SACKEG	SAMLS	SACSEN	SAWCOL	SATRF
SPIAD	D7	D6	D5	D4	D3	D2	D1	D0

SPIA Registers List

- **SPIAD 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 SPIA 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

There are also two control registers for the SPIA interface, SPIAC0 and SPIAC1. Register SPIAC0 is used to control the enable/disable function and to set the data transmission clock frequency. Register SPIAC1 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	SASPIA1	SASPIA0	—	—	—	SPIAEN	—
R/W	R/W	R/W	R/W	—	—	—	R/W	—
POR	1	1	1	—	—	—	0	—

Bit 7~5 **SASPI2~SASPI0**: SPIA Master/Slave clock select
000: SPIA master mode with clock f_{SYS} /4
001: SPIA master mode with clock f_{SYS} /16
010: SPIA master mode with clock f_{SYS} /64
011: SPIA master mode with clock f_{TBC}
100: SPIA master mode with clock CTM0 CCRP match frequency/2
101: SPIA slave mode
11x: Reserved

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 the SPI 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 Unimplemented, read as "0"

• **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 is high base level and data capture at SCKA rising edge
1: SCKA is high base level and data capture at SCKA falling edge
SACKPOLB=1
0: SCKA is low base level and data capture at SCKA falling edge
1: SCKA is low base level and data capture at 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 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 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 function will be disabled and can be placed into I/O pin or other pin-shared functions. If the bit is high, the $\overline{\text{SCSA}}$ pin will be enabled and used as a select pin.

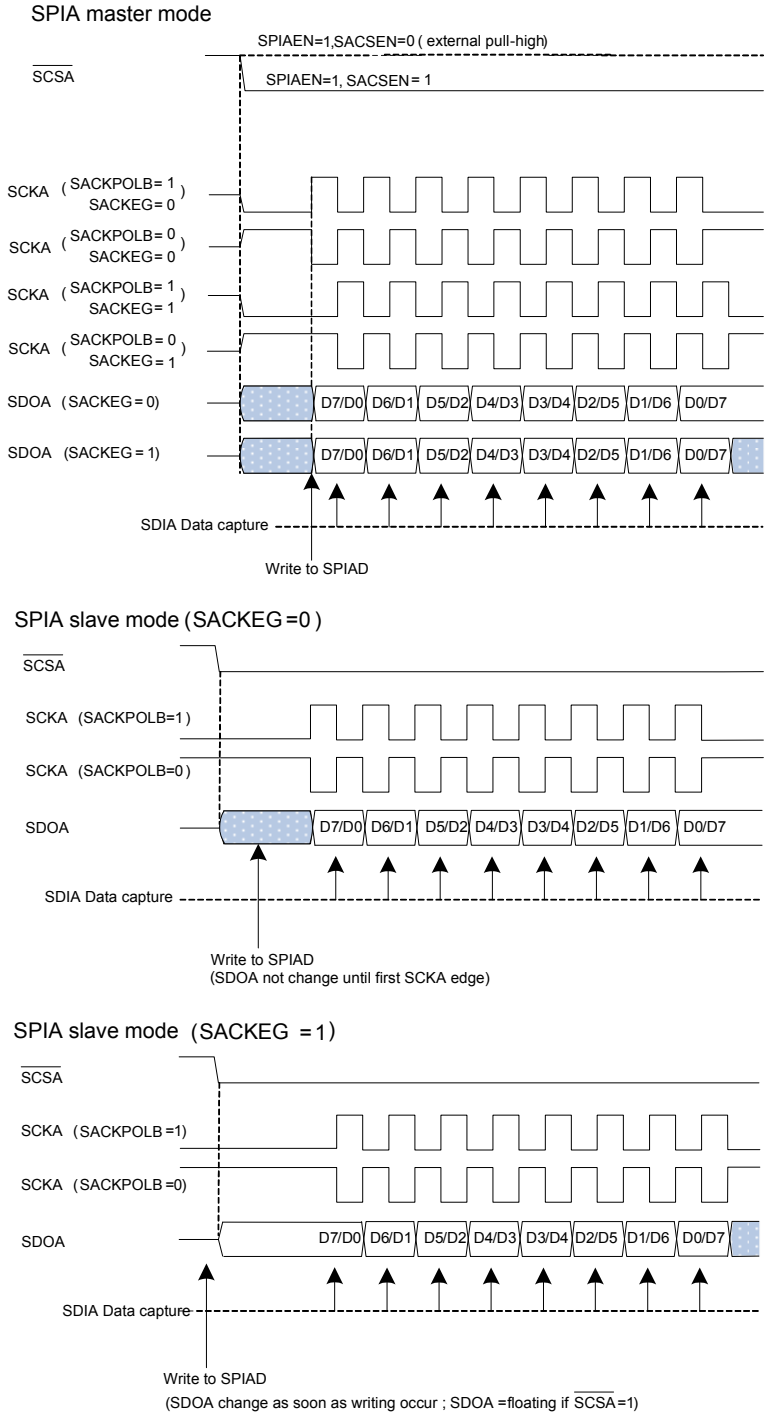
Bit 1 **SAWCOL**: SPIA write collision flag
0: No collision
1: Collision
The SAWCOL flag is used to detect whether a data collision has occurred or not. If this bit is high, it means that data has been attempted to be written to the SPIAD register duting a data transfer operation. This writing operation will be ignored if data is being transferred. This bit can be cleared by the application program.

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

SPIA Communication

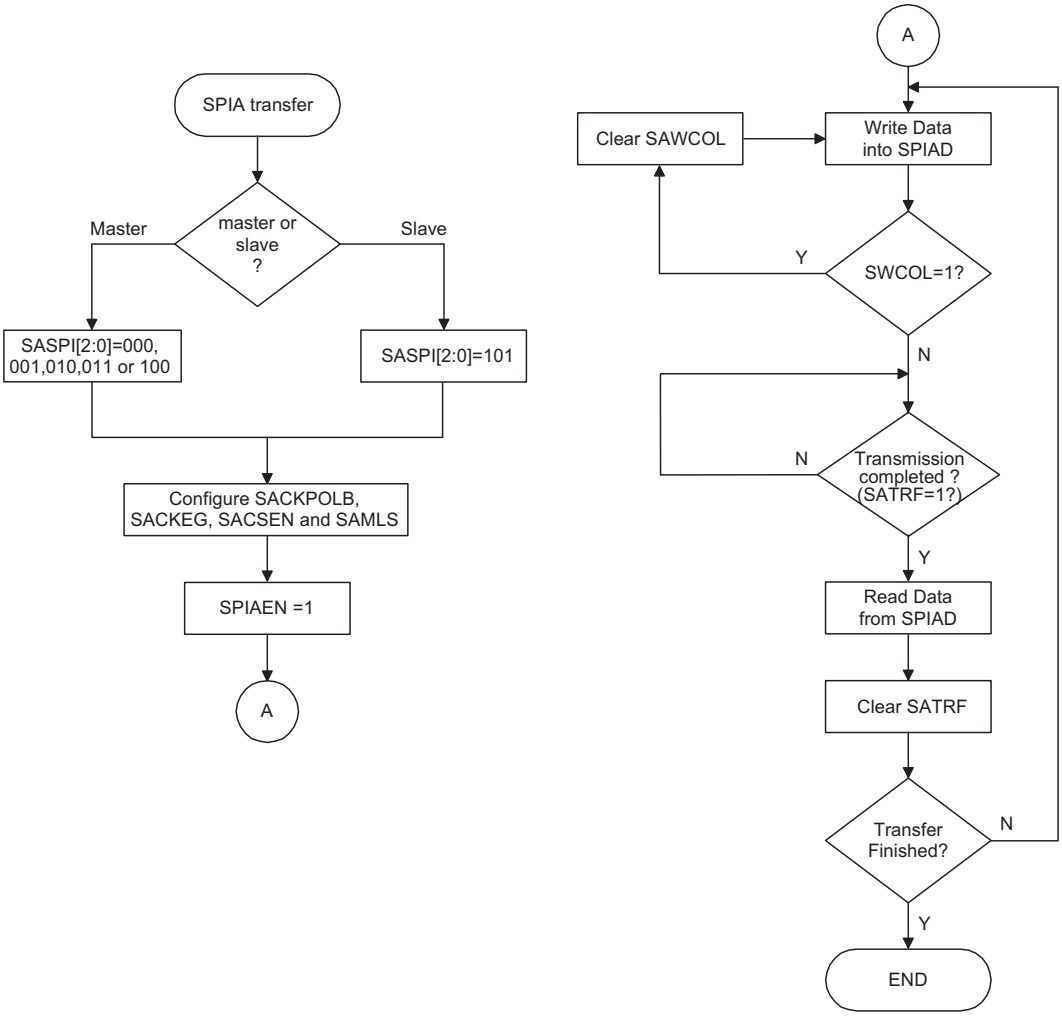
After the SPIA interface is enabled by setting the SPIAEN bit high, then in the Master Mode, when data is written to the SPIAD register, transmission/reception will begin simultaneously. When the data transfer is complete, the SATRF flag will be set automatically, but must be cleared using the application 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 registers.

The master should output a $\overline{\text{SCSA}}$ 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 $\overline{\text{SCSA}}$ signal depending upon the configurations of the SACKPOLB bit and SACKEG bit. The accompanying timing diagram shows the relationship between the slave data and $\overline{\text{SCSA}}$ signal for various configurations of the SACKPOLB and SACKEG bits. The SPIA will continue to function if the SPIA clock source is active.



Note: For SPIA slave mode, if SPIAEN=1 and SACSEN=0, SPIA is always enabled and ignore the $\overline{SCSA}$ level.

SPIA Master/Slave Mode Timing Diagram



SPIA Transfer Control Flow Chart

SPIA Bus Enable/Disable

To enable the SPIA bus, set SACSEN=1 and $\overline{SCSA}$ =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, the SCKA, SDIA, SDOA and $\overline{SCSA}$ pins can become I/O pins or other pin-shared functions using the corresponding pin-shared function control bits.

SPIA Operation

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 $\overline{SCSA}$ 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 $\overline{SCSA}$ line will be an I/O pin or other pin-shared functions 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 $\overline{SCSA}$, SDIA, SDOA and SCKA pins can be used as I/O pins or other pin-shared functions. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the 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 shifted 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 $\overline{SCSA}$ 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 SPI 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 shifted 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.

LCD Driver

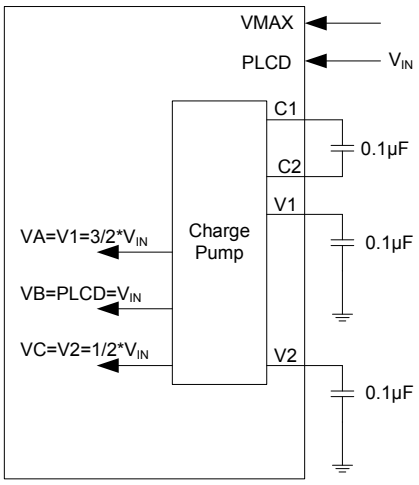
For large volume applications, which incorporate an LCD in their design, the use of a custom display rather than a more expensive character based display reduces costs significantly. However, the corresponding COM and SEG signals required, which vary in both amplitude and time, to drive such a custom display require many special considerations for proper LCD operation to occur. This series of devices all 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.

All devices include a wide range of options to enable LCD displays of various types to be driven. The table shows the range of options available across the device range.

Device	Duty	Driver Output	Bias	Bias Type	Wave Type
HT67F60A	1/4	56x4	1/3	C or R	A or B
HT67F70A					

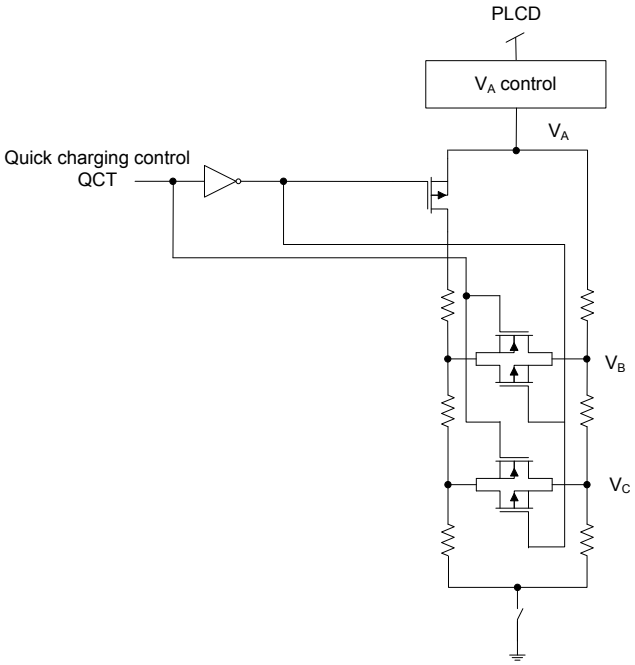
Note: For 48 LQFP package type, only the R type bias cab be used.

LCD Selections

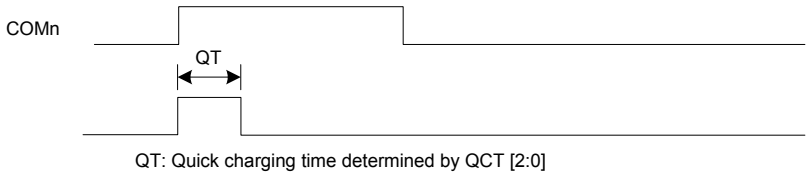


Note: The pin VMAX must be connected to the maximum voltage in this device to prevent from the pad leakage.

C Type Bias Power Supply Configuration – 1/3 Bias



Note: When the R type LCD is disabled, the DC path will be switched.



QT: Quick charging time determined by QCT [2:0]

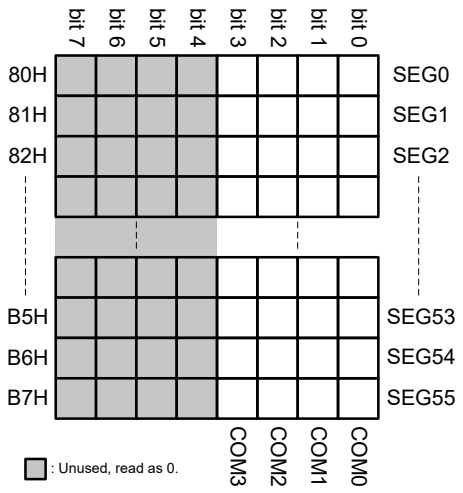
R Type Bias Configuration – 1/3 Bias

LCD 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.

As the LCD Memory addresses overlap those of the General Purpose Data Memory, it s stored in its own independent Sector 1 area. The Data Memory Sector to be used is chosen by using the Memory Pointer high byte register, which is a special function register in the Data Memory, with the name, MP1H or MP2H. To access the LCD Memory therefore requires first that Sector 1 is selected by writing a value of 01H to the MP1H or MP2H register. After this, the memory can then be accessed by using indirect addressing through the use of Memory Pointer low byte MP1L or MP2L. With Sector 1 selected, then using MP1L or MP2L to read or write to the memory area, starting with address “80H” for all the devices, will result in operations to the LCD Memory. Directly addressing the Display Memory is carried out using the corresponding extended instructions.

The accompanying LCD Memory Map diagrams shows how the internal LCD Memory is mapped to the Segments and Commons of the display for the devices. LCD Memory Maps for devices with smaller memory capacities can be extrapolated from these diagrams.



LCD Memory Map

LCD Clock Source

The LCD clock source is derived from the internal clock signal, f_{SUB}. The f_{SUB} internal clock is supplied by either the LIRC or LXT oscillator, the choice of which is determined by a configuration option. For proper LCD operation, the LCD clock source, f_{SUB}, will be internally divided by 8 using the LCD internal divider circuit to generate an ideal LCD clock source frequency of 4 kHz.

LCD Registers

There are control registers, named as LCDC0 and LCDC1, in the Data Memory used to control the various setup features of the LCD Driver.

Various bits in these registers control functions such as LCD wave type, bias type, bias resistor selection as well as overall LCD enable and disable control.

The LCDEN bit in the LCDC0 register, which provides the overall LCD enable/disable function, will only be effective when the device is in the NOAMRL, SLOW or IDLE Mode. If the device is in the SLEEP Mode then the display will always be disabled. Bits, RSEL2 ~ RSEL0, in the LCDC0 register are used to select the internal bias resistors to supply the LCD panel with the proper R type bias current. A choice to best match the LCD panel used in the application can be selected also to minimise bias current. 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 same register is used to select whether R Type or C Type LCD bias is used.

The PLCD3~PLCD0 bits in the LCDC1 register are used to select the V_A voltage for R type bias circuitry. The QCT2~QCT0 bits in the same register are used to determine the quick charge time period.

Register Name	Bit							
	7	6	5	4	3	2	1	0
LCDC0	TYPE	RCT	—	—	RSEL2	RSEL1	RSEL0	LCDEN
LCDC1	QCT2	QCT1	QCT0	—	PLCD3	PLCD2	PLCD1	PLCD0

LCD Registers List

LCDC0 Register

Bit	7	6	5	4	3	2	1	0
Name	TYPE	RCT	—	—	RSEL2	RSEL1	RSEL0	LCDEN
R/W	R/W	R/W	—	—	R/W	R/W	R/W	R/W
POR	0	0	—	—	0	0	0	0

Bit 7 **TYPE**: LCD waveform type selection
0: Type A
1: Type B

Bit 6 **RCT**: LCD bias type selection
0: R type bias
1: C type bias

Bit 5~4 Unimplemented, read as "0"

Bit 3~1 **RSEL2~RSEL0**: R type bias resistor selection
000: 1170 kΩ
001: 225 kΩ
010: 60 kΩ
011: Quick charge mode – switched between 60 kΩ and 1170 kΩ
1xx: Quick charge mode – switched between 60 kΩ and 225 kΩ

Bit 0 **LCDEN**: LCD function enable control
0: Disable
1: Enable
In the NORMAL, SLOW or IDLE mode, the LCD on/off function can be controlled by this bit. however, in the SLEEP mode, the LCD function is always switched off.

LCDC1 Register

Bit	7	6	5	4	3	2	1	0
Name	QCT2	QCT1	QCT0	—	PLCD3	PLCD2	PLCD1	PLCD0
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 **QCT2~QCT0**: R type bias Quick charge time period selection
000: 1 t_{SUB}
001: 2 t_{SUB}
010: 3 t_{SUB}
011: 4 t_{SUB}
100: 5 t_{SUB}
101: 6 t_{SUB}
110: 7 t_{SUB}
111: 8 t_{SUB}
Note that the t_{SUB} is the period of the LCD clock source, f_{SUB}.

Bit 4 Unimplemented, read as "0"

Bit 3~0 **PLCD3~PLCD0**: R type bias V_A voltage selection
0000: 8/16 × V_{PLCD}
0001: 9/16 × V_{PLCD}
0010: 10/16 × V_{PLCD}
0011: 11/16 × V_{PLCD}
0100: 12/16 × V_{PLCD}
0101: 13/16 × V_{PLCD}
0110: 14/16 × V_{PLCD}
0111: 15/16 × V_{PLCD}
1xxx: V_{PLCD}

LCD Voltage Source 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 named RCT. Selecting the C type biasing will enable an internal charge pump circuitry.

R Type Biasing

For R type biasing an external LCD voltage source must be supplied on pin PLCD to generate the internal biasing voltages. This could be the microcontroller power supply or some other voltage source. 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 selected by the PLCD 3~PLCD0 bits to be equal to a specific ratio of V_{PLCD} varying from 8/16 V_{PLCD} to V_{PLCD}. The voltage V_B is equal to $V_{PLCD}\times2/3$ while the voltage V_C is equal to $V_{PLCD}\times1/3$.

Different values of internal bias resistors can be selected using the RSEL2~RESEL0 bits in the LCDC0 register. This along with the voltage on pin PLCD will determine the bias current. The connection to the VMAX pin depends upon the voltage that is applied to the PLCD pin. If the V_{DD} voltage is greater than or equal to the voltage applied to the PLCD pin then the VMAX pin should be connected to VDD. Note that for R type biasing the voltage on the PLCD pin should not be greater than the VDD pin voltage. Note that no external capacitors or resistors are required to be connected if R type biasing is used.

Condition	VMAX Connection
$V_{DD} \geq V_{PLCD}$	Connect VMAX to VDD
$V_{DD} < V_{PLCD}$	Forbidden condition

R Type Bias VMAX Pin Connection

C Type Biasing

For C type biasing an external LCD voltage source is supplied on the PLCD pin to generate the internal biasing voltages. The C type biasing scheme uses an internal charge pump circuit can generate voltages higher than what is supplied on the PLCD or V2 pin. This feature is useful in applications where the microcontroller supply voltage is less than the supply voltage required by the LCD. An additional charge pump capacitor must also be connected between pins C1 and C2 to generate the necessary voltage levels.

For the C type 1/3 bias scheme, four voltage levels V_{SS}, V_A, V_B and V_C are utilised. The voltage V_A is generated internally and has a value of $V_{PLCD}\times3/2$. The voltage V_B will have a value equal to V_{PLCD} and V_C will have a value equal to $V_{PLCD}\times1/2$. The connection to the VMAX pin depends upon the bias and the voltage that is applied to PLCD. 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
$V_{DD} > V_{PLCD} \times 1.5$	Connect VMAX to VDD
Otherwise	Connect VMAX to V1

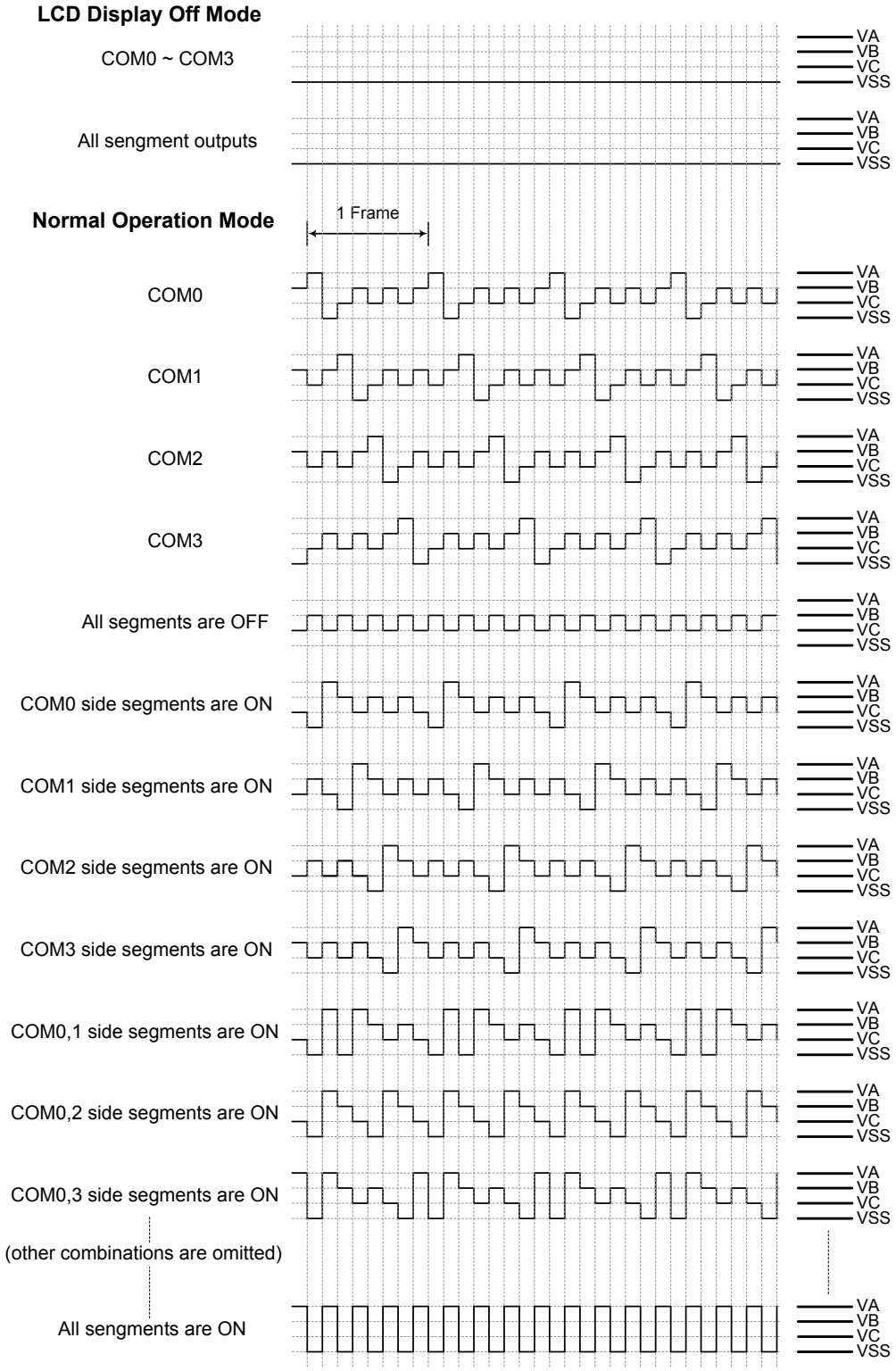
C Type Bias VMAX Pin Connection

LCD Driver Output

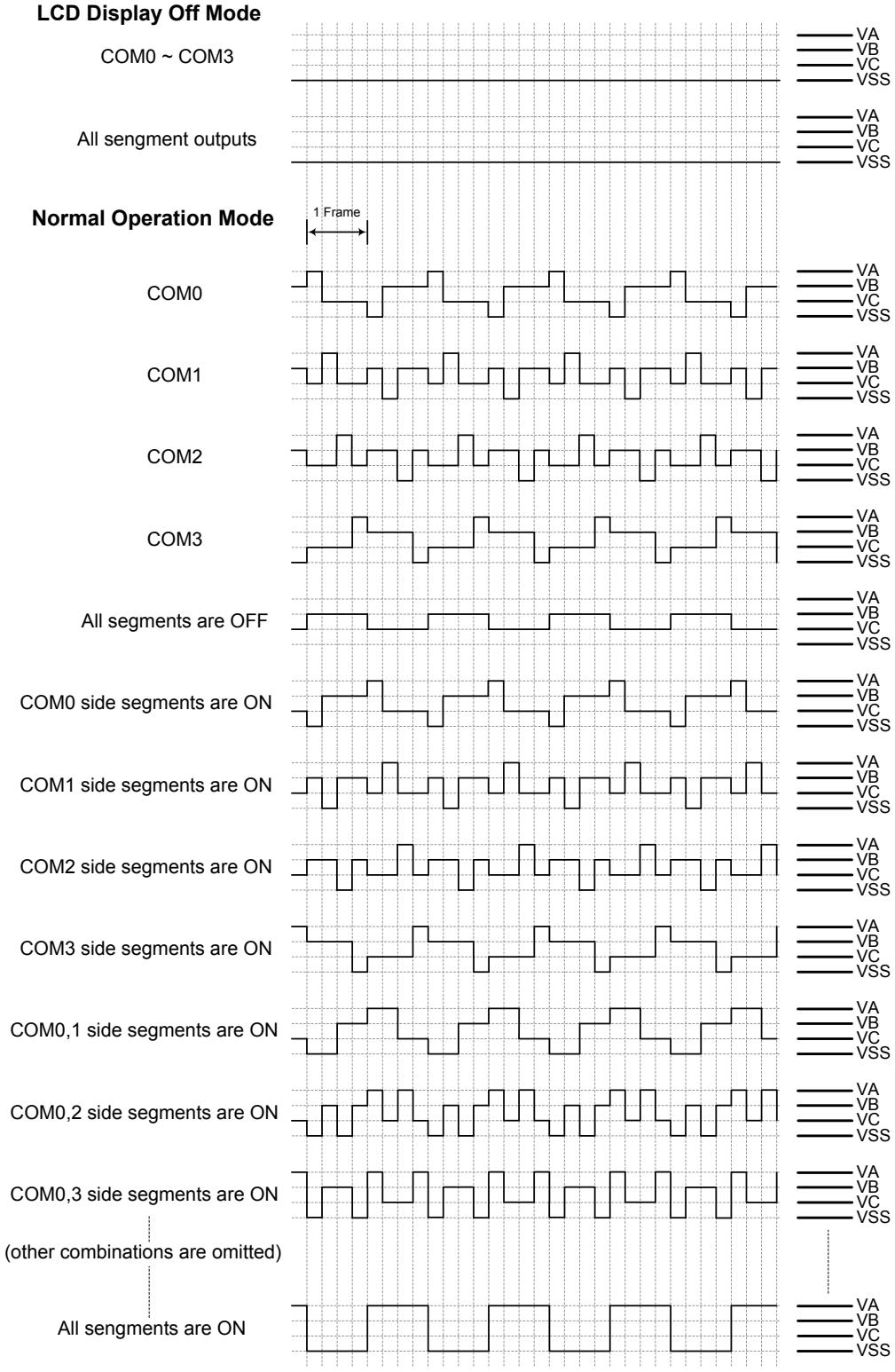
The number of COM and SEG outputs supplied by the LCD driver, as well as its biasing and wave type selections, are dependent upon how the LCD control bits are programmed. The Bias Type, whether C or R type is also selected by a software control bit.

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. The duty, which has a value of 1/4 and which 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.



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



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

Note: For 1/3 R type bias, $V_A=V_{PLCD}$, $V_B=V_{PLCD} \times 2/3$, $V_C=V_{PLCD} \times 1/3$.
For 1/3 C type bias, $V_A=V_{PLCD} \times 3/2$, $V_B=V_{PLCD}$, $V_C=V_{PLCD} \times 1/2$.

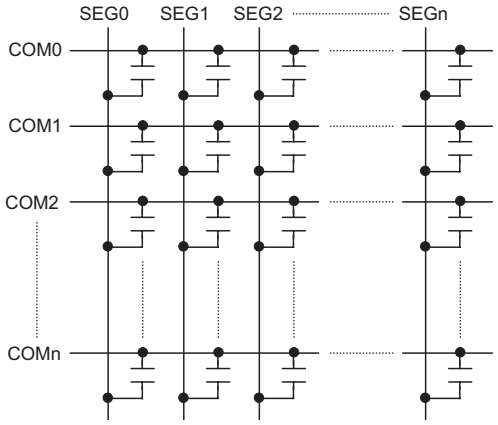
Programming Considerations

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 will be cleared to zero, the display function will be disabled.



LCD Panel Euqivalent Circuit

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions. The external interrupts are generated by the action of the external INT0 ~ INT3 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, LVD, EEPROM, SIM 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 depends upon the device chosen but fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI4 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 interrupts 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 Pins	INTnE	INTnF	n = 0 ~ 3
comparator	CPnE	CPnF	n = 0 ~ 1
A/D Converter	ADE	ADF	—
Time Base	TBnE	TBnF	n = 0 ~ 1
Multi-function	MFnE	MFnF	n = 0 ~ 4
LVD	LVE	LVF	—
EEPROM write operation	DEE	DEF	—
SIM	SIME	SIMF	—
Peripheral Interrupt $\overline{\text{PINT}}$	XPE	XPF	—
SPIA	SPIAE	SPIAF	—
CTM	CTMnPE	CTMnPF	n = 0 ~ 1
	CTMnAE	CTMnAF	
STM	STMnPE	STMnPF	n = 0 ~ 2
	STMnAE	STMnAF	
ETM	ETMPE	ETMPF	—
	ETMAE	ETMAF	
	ETMBE	ETMBF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	CP0F	INT1F	INT0F	CP0E	INT1E	INT0E	EMI
INTC1	ADF	MF1F	MF0F	CP1F	ADE	MF1E	MF0E	CP1E
INTC2	MF3F	TB1F	TB0F	MF2F	MF3E	TB1E	TB0E	MF2E
INTC3	—	MF4F	INT3F	INT2F	—	MF4E	INT3E	INT2E
MFI0	STM0AF	STM0PF	CTM0AF	CTM0PF	STM0AE	STM0PE	CTM0AE	CTM0PE
MFI1	—	ETMBF	ETMAF	ETMPF	—	ETMBE	ETMAE	ETMPE
MFI2	SIMF	XPF	CTM1AF	CTM1PF	SIME	XPE	CTM1AE	CTM1PE
MFI3	—	SPIAF	DEF	LVF	—	SPIAE	DEE	LVE
MFI4	STM2AF	STM2PF	STM1AF	STM1PF	STM2AE	STM2PE	STM1AE	STM1PE

Interrupt Registers List

INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
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 **INT3S1~INT3S0**: Interrupt edge control for INT3 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 5~4 **INT2S1~INT2S0**: Interrupt edge control for INT2 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

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	—	CP0F	INT1F	INT0F	CP0E	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 **CP0F**: Comparator 0 interrupt request flag
0: No request
1: Interrupt request

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

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

Bit 3 **CP0E**: Comparator 0 interrupt control
0: Disable
1: Enable

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

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

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

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	ADF	MF1F	MF0F	CP1F	ADE	MF1E	MF0E	CP1E
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 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

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

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

Bit 4 **CP1F**: Comparator 1 interrupt request flag
0: No request
1: Interrupt request

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

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

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

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

INTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	MF3F	TB1F	TB0F	MF2F	MF3E	TB1E	TB0E	MF2E
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 **MF3F**: Multi-function 3 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **MF2F**: Multi-function 2 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **MF3E**: Multi-function 3 interrupt control
0: Disable
1: Enable

Bit 2 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

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

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

INTC3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	MF4F	INT3F	INT2F	—	MF4E	INT3E	INT2E
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 **MF4F**: Multi-function 4 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **INT3F**: INT3 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INT2F**: INT2 interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **MF4E**: Multi-function 4 interrupt control
0: Disable
1: Enable

Bit 1 **INT3E**: INT3 interrupt control
0: Disable
1: Enable

Bit 0 **INT2E**: INT2 interrupt control
0: Disable
1: Enable

MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	STM0AF	STM0PF	CTM0AF	CTM0PF	STM0AE	STM0PE	CTM0AE	CTM0PE
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 **STM0AF**: STM0 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 6 **STM0PF**: STM0 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

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

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

Bit 3 **STM0AE**: STM0 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 2 **STM0PE**: STM0 Comparator P match Interrupt control
0: Disable
1: Enable

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

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

MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	ETMBF	ETMAF	ETMPF	—	ETMBE	ETMAE	ETMPE
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 **ETMBF**: ETM Comparator B match Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **ETMAF**: ETM Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **ETMPF**: ETM Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **ETMAE**: ETM Comparator B match Interrupt control
0: Disable
1: Enable

Bit 1 **ETMAE**: ETM Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **ETMPE**: ETM Comparator P match Interrupt control
0: Disable
1: Enable

MFI2 Register

Bit	7	6	5	4	3	2	1	0
Name	SIMF	XPF	CTM1AF	CTM1PF	SIME	XPE	CTM1AE	CTM1PE
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 **SIMF**: SIM Interrupt request flag
0: No request
1: Interrupt request

Bit 6 **XPF**: External peripheral Interrupt request flag
0: No request
1: Interrupt request

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

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

Bit 3 **SIME**: SIM Interrupt control
0: Disable
1: Enable

Bit 2 **XPE**: External peripheral Interrupt control
0: Disable
1: Enable

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

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

MFI3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	SPIAF	DEF	LVF	—	SPIAE	DEE	LVE
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 **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 Unimplemented, read as "0"

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

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

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

MFI4 Register

Bit	7	6	5	4	3	2	1	0
Name	STM2AF	STM2PF	STM1AF	STM1PF	STM2AE	STM2PE	STM1AE	STM1PE
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 **STM2AF**: STM2 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 6 **STM2PF**: STM2 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **STM1AF**: STM1 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **STM1PF**: STM1 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3 **STM2AE**: STM2 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 2 **STM2PE**: STM2 Comparator P match Interrupt control
0: Disable
1: Enable

Bit 1 **STM1AE**: STM1 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **STM1PE**: STM1 Comparator P match Interrupt control
0: Disable
1: Enable

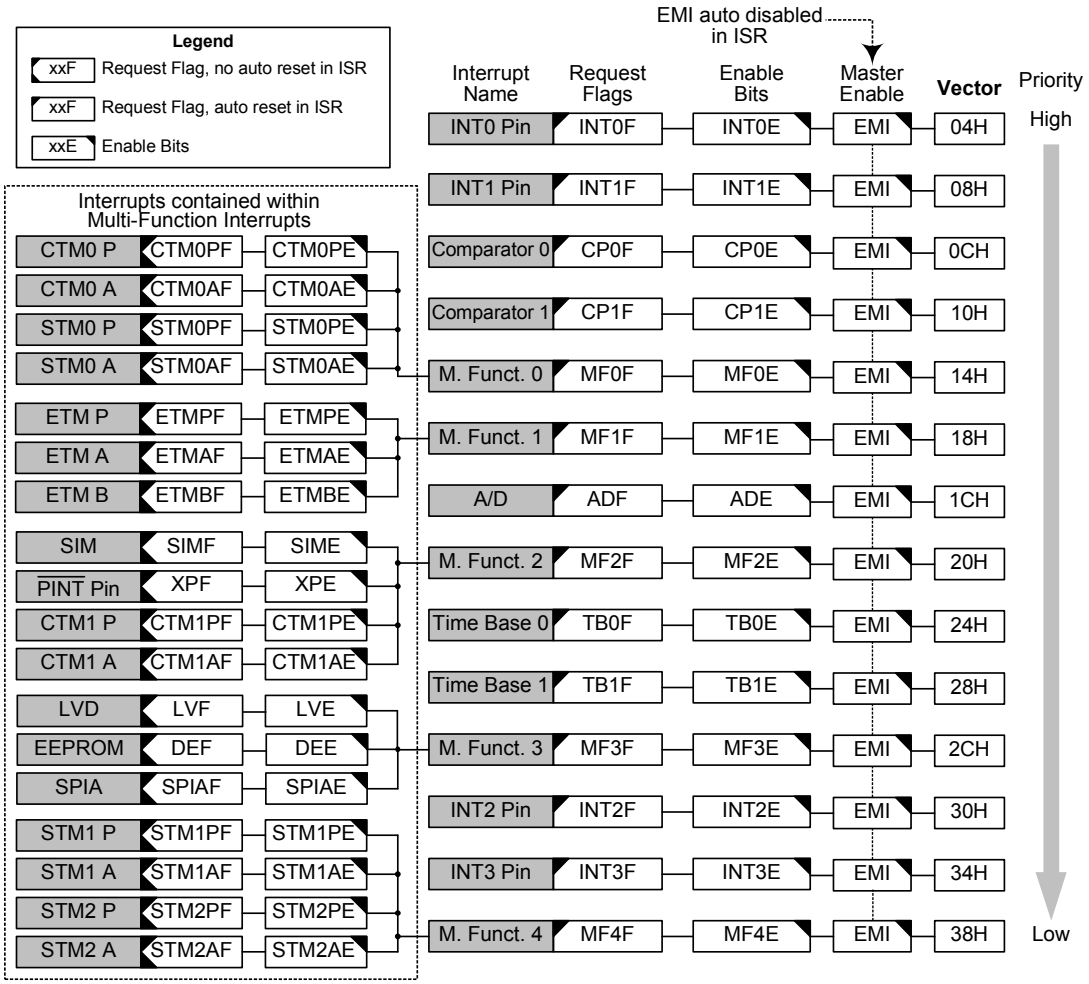
Interrupt Operation

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

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

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.



Interrupt Scheme

External Interrupt

The external interrupts are controlled by signal transitions on the pins INT0~INT3. An external interrupt request will take place when the external interrupt request flags, INT0F~INT3F, 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~INT3E, 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. The pin must also be setup as an input by setting the corresponding bit in the port control register as well as the relevant pin-shared function selection bits. 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~INT3F, 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.

Comparator Interrupt

The comparator interrupts are controlled by the internal comparators. A comparator interrupt request will take place when the comparator interrupt request flags, CP0F or CP1F, are set, a situation that will occur when the comparator output changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and comparator interrupt enable bits, CP0E and CP1E, must first be set. When the interrupt is enabled, the stack is not full and the voice play timer time-out occurs, a subroutine call to the comparator interrupt vector will take place. When the interrupt is serviced, the comparator interrupt request flag will be automatically reset and the EMI bit will also be automatically cleared to disable other interrupts.

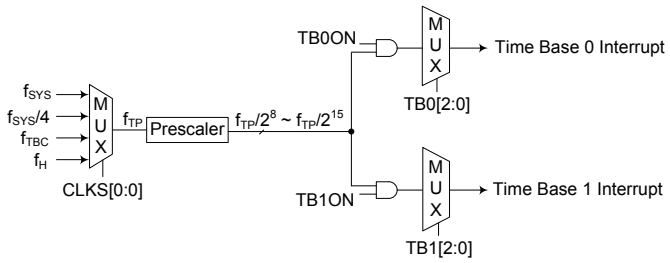
A/D Converter Interrupt

The A/D Converter Interrupt is controlled by the termination of an A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter Interrupt vector, will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Time Base Interrupt

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

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, f_{TP}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$, f_{SUB} or f_H 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 CLKS01 and CLKS00 bits in the PSC0 register.



Time Base Interrupts

PSC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKS01	CLKS00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKS01~CLKS00**: Time Base prescaler clock source selection
00: f_{SYS}
01: $f_{SYS}/4$
10: f_{TBC}
11: f_H

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 Enable Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

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

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 Enable Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

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

Multi-function Interrupt

Within the device there are up to five Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts, LVD interrupt, EEPROM write operation interrupt, SIM, External peripheral interrupt and SPIA interface interrupts.

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 request flags will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupts will not be automatically reset and must be manually reset by the application program.

Serial Interface Module Interrupt

The Serial Interface Module Interrupt, also known as the SIM interrupt, is contained within the Multi-function Interrupt. A SIM Interrupt request will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, the Serial Interface Interrupt enable bit, SIME, and Muti-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SIM interface, a subroutine call to the respective Multi-function Interrupt vector, will take place. When the Serial Interface 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 SIMF flag will not be automatically cleared, it has to be cleared by the application program.

SPIA Interface Interrupt

The SPIA Interface Module Interrupt is contained within the Multi-function Interrupt. A SPIA Interrupt request 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, the Serial Interface Interrupt enable bit, SPIAE, and Muti-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SPIA interface, a subroutine call to the respective Multi-function Interrupt vector, will take place. When the SPIA Interface 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 SPIAF flag will not be automatically cleared, it has to be cleared by the application program.

External Peripheral Interrupt

The External Peripheral Interrupt operates in a similar way to the external interrupt and is contained within the Multi-function Interrupt. A Peripheral Interrupt request will take place when the External Peripheral Interrupt request flag, XPF, is set, which occurs when a negative edge transition appears on the $\overline{\text{PINT}}$ pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, external peripheral interrupt enable bit, XPE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a negative transition appears on the External Peripheral Interrupt pin, a subroutine call to the respective Multi-function Interrupt, will take place. When the External Peripheral 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 XPF flag will not be automatically cleared, it has to be cleared by the application program. The external peripheral interrupt pin is pin-shared with several other pins with different functions. It must therefore be properly configured to enable it to operate as an External Peripheral Interrupt pin.

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. 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.

EEPROM Interrupt

The EEPROM Write Interrupt is contained within the Multi-function Interrupt. An EEPROM Write Interrupt request will take place when the EEPROM Write Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, EEPROM Write Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the respective Multi-function Interrupt vector will take place. When the EEPROM Write Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the Multi-function interrupt request flag will be automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

TM Interrupt

The Compact, Standard and Enhanced TMs have two or three interrupts, each comes from the comparator A match situation, comparator B match situation or comparator P match situation respectively. 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, comparator A or comparator B 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 these devices are in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins, a low power supply voltage or comparator input change 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.

Low Voltage Detector – LVD

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

LVD Register

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

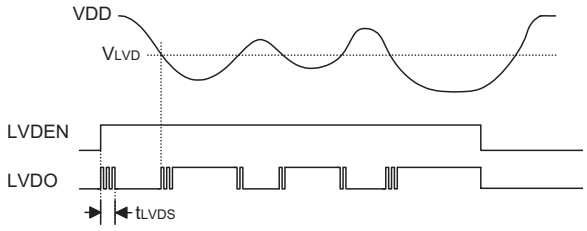
Bit 4 **LVDEN**: Low Voltage Detector Enable control
0: Disable
1: Enable

Bit 3 **VBGEN**: Bandgap buffer 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 to 1.

Bit 2~0 **VLVD2~VLVD0**: LVD Voltage selection
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.0V 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. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is powered down the low voltage detector will remain active 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} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

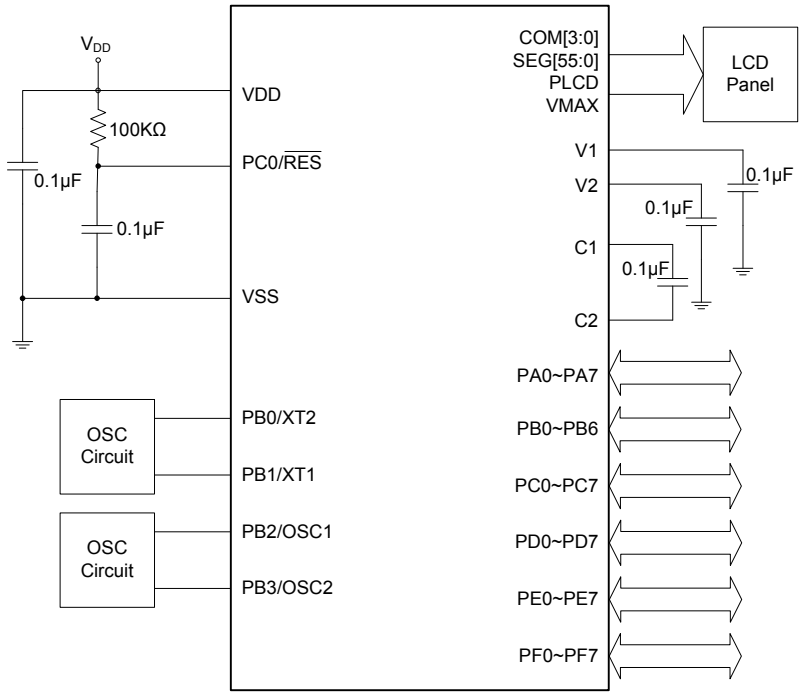
The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set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

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the devices 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 devices 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
1	High Speed System Oscillator Selection – f_H HXT or HIRC
2	Low Speed System Oscillator Selection – f_{SUB} LXT or LIRC
3	High Speed Internal RC Oscillator Frequency Selection – f_{HIRC} 4MHz, 8MHz or 12MHz

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of several kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

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

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

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

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

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	2Note	Z, C, AC, OV, SC
LADC A,[m]	Add Data Memory to ACC with Carry	2	Z, C, AC, OV, SC
LADCM A,[m]	Add ACC to Data memory with Carry	2Note	Z, C, AC, OV, SC
LSUB A,[m]	Subtract Data Memory from ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LSBC A,[m]	Subtract Data Memory from ACC with Carry	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2Note	C
Logic Operation			
LAND A,[m]	Logical AND Data Memory to ACC	2	Z
LOR A,[m]	Logical OR Data Memory to ACC	2	Z
LXOR A,[m]	Logical XOR Data Memory to ACC	2	Z
LANDM A,[m]	Logical AND ACC to Data Memory	2Note	Z
LORM A,[m]	Logical OR ACC to Data Memory	2Note	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2Note	Z
LCPL [m]	Complement Data Memory	2Note	Z
LCPLA [m]	Complement Data Memory with result in ACC	2	Z
Increment & Decrement			
LINCA [m]	Increment Data Memory with result in ACC	2	Z
LINC [m]	Increment Data Memory	2Note	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2Note	Z
Rotate			
LRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2Note	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2Note	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2Note	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2Note	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2Note	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2Note	None
LSET [m].i	Set bit of Data Memory	2Note	None

Mnemonic	Description	Cycles	Flag Affected
Branch			
LSZ [m]	Skip if Data Memory is zero	2Note	None
LSZA [m]	Skip if Data Memory is zero with data movement to ACC	2Note	None
LSNZ [m]	Skip if Data Memory is not zero	2Note	None
LSZ [m].i	Skip if bit i of Data Memory is zero	2Note	None
LSNZ [m].i	Skip if bit i of Data Memory is not zero	2Note	None
LSIZ [m]	Skip if increment Data Memory is zero	2Note	None
LSDZ [m]	Skip if decrement Data Memory is zero	2Note	None
LSIZA [m]	Skip if increment Data Memory is zero with result in ACC	2Note	None
LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC	2Note	None
Table Read			
LTABRD [m]	Read table (specific page) to TBLH and Data Memory	3Note	None
LTABRDL [m]	Read table (last page) to TBLH and Data Memory	3Note	None
LITABRD [m]	Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory	3Note	None
LITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	3Note	None
Miscellaneous			
LCLR [m]	Clear Data Memory	2Note	None
LSET [m]	Set Data Memory	2Note	None
LSWAP [m]	Swap nibbles of Data Memory	2Note	None
LSWAPA [m]	Swap nibbles of Data Memory with result in ACC	2	None

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

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

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

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

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

Affected flag(s): None

CLR [m] Clear Data Memory

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

Operation: [m] ← 00H

Affected flag(s): None

CLR [m].i Clear bit of Data Memory

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

Operation: [m].i ← 0

Affected flag(s): None

CLR WDT Clear Watchdog Timer

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

Operation: WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CPL [m] Complement Data Memory

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

Operation: [m] ← $\overline{[m]}$

Affected flag(s): Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation: ACC ← $\overline{[m]}$

Affected flag(s): Z

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

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

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

Affected flag(s): C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC "OR" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − $\overline{C}$

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

SBC A, x Subtract immediate data from ACC with Carry

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

Operation ACC ← ACC – [m] – $\overline{C}$

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

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

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

Operation [m] ← ACC − [m] − $\overline{C}$

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

SDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

SNZ [m].i Skip if bit i of Data Memory is not 0

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

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

Operation Skip if [m]≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

TABRD [m]	Read table (specific page) to TBLH and Data Memory
Description	The low byte of the program code (specific page) addressed by the table pointer (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

TABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

ITABRD [m]	Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

ITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

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

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

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

Extended Instruction Definition

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

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

LADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

LADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

LCLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

LCLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

LCPL [m] Complement Data Memory

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

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

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 three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

LSET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

LSET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

LSIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

LSIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

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

Operation Skip if [m] ≠ 0

Affected flag(s) None

LSUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

LSUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

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

Operation [m] ← ACC − [m]

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

LSWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

LSZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

Description The low byte of the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

LTABRDL [m] Read table (last page) to TBLH and Data Memory

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

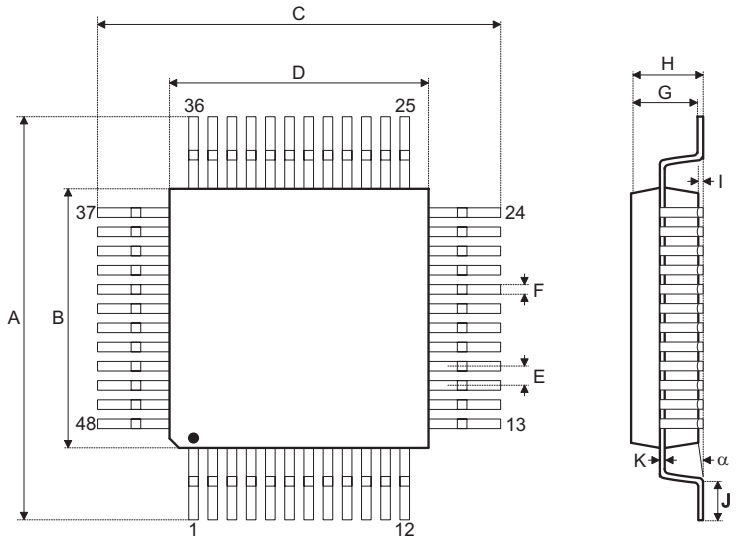
Package Information

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

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

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- Carton information

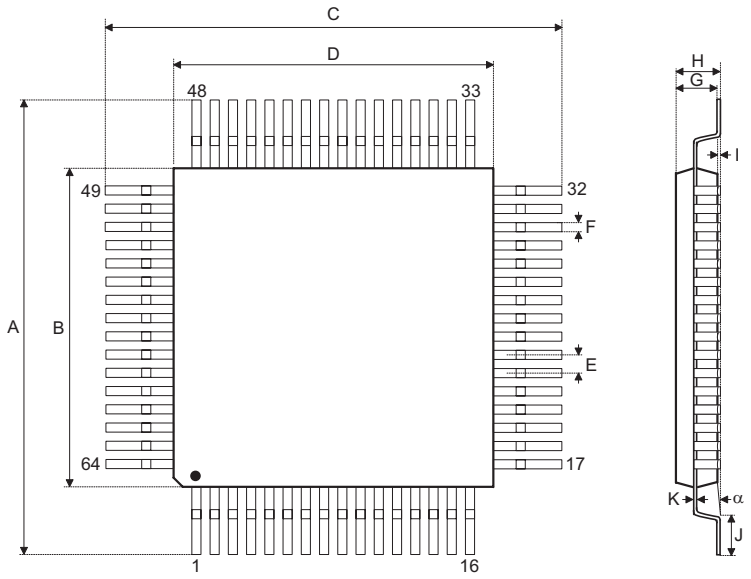
48-pin LQFP (7mm×7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.020 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	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.50 BSC	—
F	0.17	0.22	0.27
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°

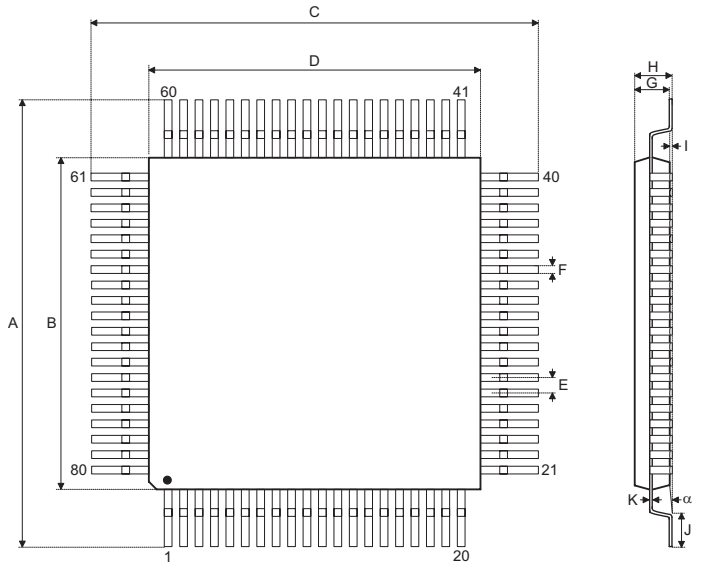
64-pin LQFP (7mm×7mm) Outline Dimensions



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	—	0.016 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	—	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°