



24-bit Delta Sigma A/D Flash MCU
Integrated Regulator & 2 set OPAs & LCD Driver

BH67F2752

Revision: V1.10 Date: December 02, 2019

www.holtek.com

Table of Contents

Features	7
CPU Features	7
Peripheral Features.....	7
General Description.....	8
Block Diagram.....	9
Pin Assignment.....	10
Pin Description	11
Absolute Maximum Ratings.....	13
D.C. Characteristics.....	13
Operating Voltage Characteristics.....	13
Operating Current Characteristics.....	13
Standby Current Characteristics	14
A.C. Characteristics.....	14
High Speed Internal Oscillator – HIRC – Frequency Accuracy	14
Low Speed Internal Oscillator Characteristics – LIRC	15
Operating Frequency Characteristic Curves	15
System Start Up Time Characteristics	15
Input/Output Characteristics	16
Memory Electrical Characteristics	16
LVD/LVR Electrical Characteristics	17
Analog Front End Circuit Characteristics	17
24-bit A/D Converter Electrical Characteristics	17
LCD Electrical Characteristics	20
Power-on Reset Characteristics.....	21
System Architecture	21
Clocking and Pipelining.....	21
Program Counter.....	22
Stack	23
Arithmetic and Logic Unit – ALU	23
Flash Program Memory	24
Structure.....	24
Special Vectors	24
Look-up Table.....	24
Table Program Example.....	25
In Circuit Programming – ICP	26
On-Chip Debug Support – OCDS	27

RAM Data Memory	27
Structure.....	27
Data Memory Addressing	28
General Purpose Data Memory	28
Special Purpose Data Memory	28
Special Function Register Description.....	30
Indirect Addressing Registers – IAR0, IAR1, IAR2	30
Memory Pointers – MP0, MP1L, MP1H, MP2L, MP2H.....	30
Accumulator – ACC.....	31
Program Counter Low Register – PCL.....	32
Look-up Table Registers – TBLP, TBHP, TBLH.....	32
Status Register – STATUS	32
EEPROM Data Memory.....	34
EEPROM Data Memory Structure	34
EEPROM Registers	34
Reading Data from the EEPROM	35
Writing Data to the EEPROM.....	36
Write Protection.....	36
EEPROM Interrupt	36
Programming Considerations.....	36
Oscillators	38
Oscillator Overview	38
System Clock Configurations	38
Internal RC Oscillator – HIRC	38
Internal 32kHz Oscillator – LIRC.....	39
Operating Modes and System Clocks	39
System Clocks	39
System Operation Modes.....	40
Control Registers	41
Operating Mode Switching	42
Standby Current Considerations	46
Wake-up	46
Watchdog Timer.....	47
Watchdog Timer Clock Source.....	47
Watchdog Timer Control Register	47
Watchdog Timer Operation	48
Reset and Initialisation.....	49
Reset Functions	49
Reset Initial Conditions	52

Input/Output Ports	55
Pull-high Resistors	55
Port A Wake-up	56
I/O Port Control Registers	56
I/O Port Source Current Control	56
Pin-shared Functions	57
I/O Pin Structures	61
Programming Considerations	61
Timer Modules – TM	62
Introduction	62
TM Operation	62
TM Clock Source	62
TM Interrupts	62
TM External Pins	62
Programming Considerations	63
Compact Type TM – CTM	64
Compact Type TM Operation	64
Compact Type TM Register Description	65
Compact Type TM Operating Modes	69
Analog to Digital Converter – ADC	75
A/D Converter Overview	75
Internal Power Supply	75
A/D Data Rate Definition	77
A/D Converter Register Description	77
A/D Operation	84
Summary of A/D Conversion Steps	85
Programming Considerations	86
A/D Transfer Function	86
A/D Converted Data	87
A/D Converted Data to Voltage	87
A/D Programming Example	88
Temperature sensor	89
UART Interface	89
UART External Pins	89
UART Data Transfer Scheme	90
UART Status and Control Registers	90
Baud Rate Generator	95
UART Setup and Control	96
UART Transmitter	97
UART Receiver	99
Managing Receiver Errors	100
UART Module Interrupt Structure	101
UART Power Down and Wake-up	102

Serial Peripheral Interface – SPI.....	103
SPI Interface Operation.....	103
SPI Registers	104
SPI Communication	107
SPI Bus Enable/Disable	109
SPI Operation.....	109
Error Detection	110
LCD Driver	111
LCD Data Memory	111
LCD Clock Source.....	112
LCD Register.....	112
LCD Voltage Source and Biasing.....	113
LCD Charge Pump.....	114
LCD Reset Status	115
LCD Driver Output.....	115
Programming Considerations.....	120
Interrupts	121
Interrupt Registers.....	121
Interrupt Operation	125
External Interrupts.....	126
LVD Interrupt.....	127
EEPROM Interrupt	127
A/D Converter Interrupt	127
Multi-function Interrupts.....	127
UART Interrupt	128
SPI Interrupt.....	128
Time Base Interrupts	128
Timer Module Interrupts	129
Interrupt Wake-up Function.....	130
Programming Considerations.....	130
Low Voltage Detector – LVD	131
LVD Register	131
LVD Operation.....	132
Application Circuits.....	133
Introduction	134
Instruction Timing	134
Moving and Transferring Data.....	134
Arithmetic Operations.....	134
Logical and Rotate Operation	135
Branches and Control Transfer	135
Bit Operations	135
Table Read Operations	135
Other Operations.....	135

Instruction Set Summary	136
Table Conventions.....	136
Extended Instruction Set.....	138
Instruction Definition.....	140
Extended Instruction Definition	149
Package Information	156
48-pin LQFP (7mm×7mm) Outline Dimensions	157
64-pin LQFP (7mm×7mm) Outline Dimensions	158

Features

CPU Features

- Operating Voltage
 - ♦ $f_{SYS}=8\text{MHz}$: 2.2V~5.5V
- Up to 0.5 μs instruction cycle with 8MHz system clock at $V_{DD}=5\text{V}$
- Power down and wake-up functions to reduce power consumption
- Oscillators
 - ♦ Internal High Speed 8MHz RC Oscillator – HIRC
 - ♦ Internal Low Speed 32kHz RC Oscillator – LIRC
- Fully integrated internal oscillators require no external components
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 8K \times 16
- RAM Data Memory: 384 \times 8
- True EEPROM Memory: 128 \times 8
- Watchdog Timer function
- Up to 17 bidirectional I/O lines
- 4 differential or 8 single-end channels 24-bit resolution Delta Sigma A/D converter
- LCD Driver function
 - ♦ SEGs \times COMs: 32 \times 4 or 30 \times 6
 - ♦ Duty type: 1/4 duty or 1/6 duty
 - ♦ Bias level: 1/3 bias
 - ♦ Bias type: R type
 - ♦ Waveform type: type A or type B
- Two pin-shared external interrupts
- Timer Modules for time measure, compare match output or PWM output function
- Fully-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- Single Serial Peripheral Interface – SPI
- Dual Time Base functions for generation of fixed time interrupt signals
- Low Voltage Reset function – LVR
- Low Voltage Detect function – LVD
- Package types: 48/64-pin LQFP

General Description

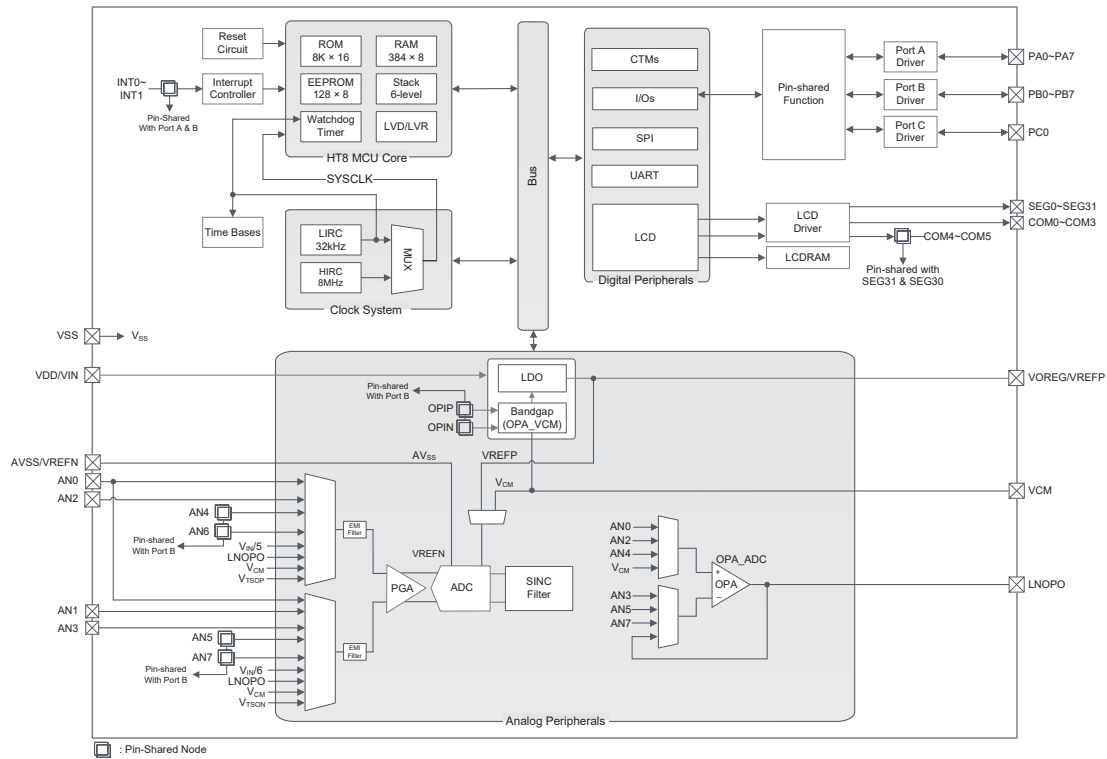
The BH67F2752 is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller which includes a multi-channel 24-bit Delta Sigma A/D converter, designed for applications that interface directly to analog signals and which require a low noise and high accuracy analog-to-digital converter. Offering users the convenience of Flash Memory multi-programming features, this device also includes 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 24-bit Delta Sigma A/D Converter and Operational Amplifier functions. Extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is implemented using fully integrated SPI and UART interfaces, popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

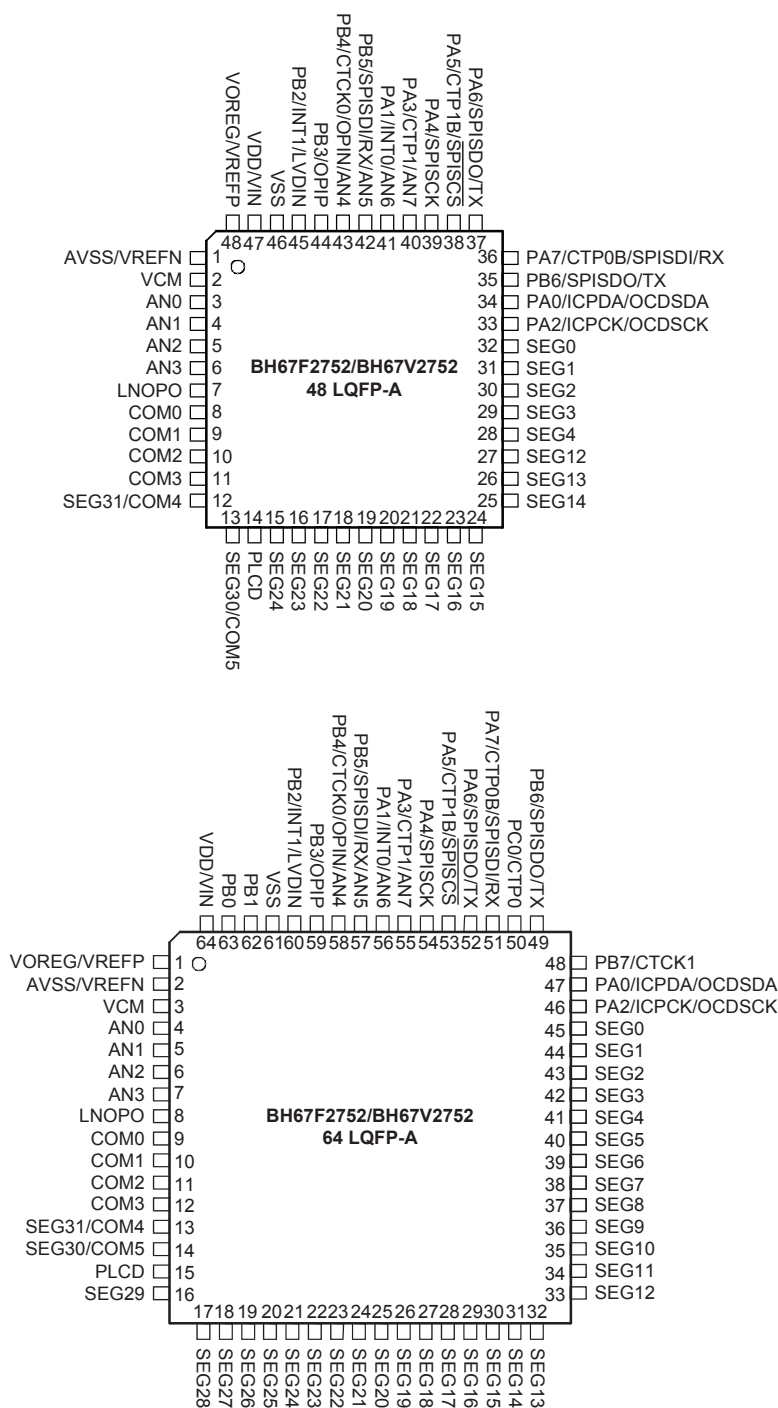
A full choice of internal low and high oscillator functions are provided including two fully integrated system oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives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

The inclusion of flexible I/O programming features, Time Base functions along with many other features ensure that the device will find excellent use in applications such as weight scales, electronic metering, environmental monitoring, handheld instruments, electronically controlled tools, motor driving in addition to many others.

Block Diagram



Pin Assignment



- Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.
2. The OCSDA and OCDSCK pins are supplied as OCDS dedicated pins and as such only available for the BH67V2752 device which is the OCDS EV chip for the BH67F2752 device.

3. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input condition. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.

Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet. Note that where more than one package type exists the table will reflect the situation for the larger package type.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/ICPDA/OCDSDA	PA0	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	ICPDA	—	ST	CMOS	ICP data/address pin
	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
PA1/INT0/AN6	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT0	PAS0 INTEG INTC0	ST	—	External interrupt 0
	AN6	PAS0	AN	—	A/D converter external analog signal input
PA2/ICPCK/OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	CMOS	OCDS clock input, for EV chip only
PA3/CTP1/AN7	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP1	PAS0	—	CMOS	CTM1 output
	AN7	PAS0	AN	—	A/D converter external analog signal input
PA4/SPISCK	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SPISCK	PAS1	ST	—	SPI serial clock
PA5/CTP1B/SPISCS	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP1B	PAS1	—	CMOS	CTM1 inverting output
	SPISCS	PAS1	ST	CMOS	SPI slave select pin
PA6/SPISDO/TX	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SPISDO	PAS1	—	CMOS	SPI serial data output
	TX	PAS1	—	CMOS	UART serial data output
PA7/CTP0B/SPISDI/ RX	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP0B	PAS1	—	CMOS	CTM0 inverting output
	SPISDI	PAS1 IFS	ST	—	SPI serial data input
	RX	PAS1 IFS	ST	—	UART serial data input
PB0~PB1	PB0~PB1	—	ST	CMOS	General purpose I/O. Register enabled pull-high

Pin Name	Function	OPT	I/T	O/T	Description
PB2/INT1/LVDIN	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	INT1	PBS0 INTEG INTC0	ST	—	External interrupt 1
	LVDIN	PBS0	AN	—	LVD input
PB3/OPIP	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	OPIP	PBS0	AN	—	OPA_VCM positive input
PB4/CTCK0/OPIN/ AN4	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTCK0	PBS1	ST	—	CTM0 clock input
	OPIN	PBS1	AN	—	OPA_VCM negative input
	AN4	PBS1	AN	—	A/D converter external analog signal input
PB5/SPISDI/RX/AN5	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SPISDI	PBS1 IFS	ST	—	SPI serial data input
	RX	PBS1 IFS	ST	—	UART serial data input
	AN5	PBS1	AN	—	A/D converter external analog signal input
PB6/SPISDO/TX	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SPISDO	PBS1	—	CMOS	SPI serial data output
	TX	PBS1	—	CMOS	UART serial data output
PB7/CTCK1	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTCK1	—	ST	—	CTM1 clock input
PC0/CTP0	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTP0	PCS0	—	CMOS	CTM0 output
VOREG/VREFP	VOREG	—	—	AN	LDO output pin
		—	AN	—	Positive power supply for ADC, PGA
	VREFP	—	AN	—	External positive reference input of ADC
AN0~AN3	ANn	—	AN	—	A/D converter external analog signal input
VCM	VCM	—	—	AN	ADC Common mode voltage output/OPA_VCM output
LNOPO	LNOPO	—	—	AN	Low noise OPA output
VDD/VIN	VDD	—	PWR	—	Positive Power supply
	VIN	—	PWR	—	LDO input pin
VSS	VSS	—	PWR	—	Negative power supply
AVSS/VREFN	AVSS	—	PWR	—	Negative power supply for VCM, ADC, PGA
	VREFN	—	AN	—	External negative reference input of ADC
PLCD	PLCD	—	PWR	AN	LCD power supply
SEG0~SEG29	SEGn	—	—	AN	LCD Segment output
SEG31/COM4	SEG31	CMOS	—	AN	LCD Segment output
	COM4	CMOS	—	AN	LCD Common output
SEG30/COM5	SEG30	CMOS	—	AN	LCD Segment output
	COM5	CMOS	—	AN	LCD Common output
COM0~COM3	COMn	—	—	AN	LCD Common output

Legend: I/T: Input type

OPT: Optional by register option

CMOS: CMOS output

PWR: Power

O/T: Output type

ST: Schmitt Trigger input

AN: Analog signal

Absolute Maximum Ratings

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

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

D.C. Characteristics

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

Operating Voltage Characteristics

$T_a=-40^{\circ}C\sim 85^{\circ}C$

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	$f_{SYS}=8MHz$	2.2	—	5.5	V
	Operating Voltage – LIRC	$f_{SYS}=32kHz$	2.2	—	5.5	V

Operating Current Characteristics

$T_a=-40^{\circ}C\sim 85^{\circ}C$

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2.2V	$f_{SYS}=32kHz$	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	2.2V	$f_{SYS}=8MHz$	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	

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

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

Standby Current Characteristics

Ta=25°C

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max.	Unit
		V _{DD}	Conditions				85°C	
I _{STB}	SLEEP Mode	2.2V	WDT off	—	0.2	0.6	0.7	μA
		3V		—	0.2	0.8	1.0	
		5V		—	0.5	1.0	1.2	
		2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3	5	6	
	IDLE0 Mode – LIRC	2.2V	f _{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE1 Mode – HIRC	2.2V	f _{SUB} on, f _{sys} =8MHz	—	288	400	480	μA
		3V		—	420	600	720	
		5V		—	800	1200	1440	

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

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

A.C. Characteristics

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

High Speed Internal Oscillator – HIRC – Frequency Accuracy

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Temp.				
f _{HIRC}	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	

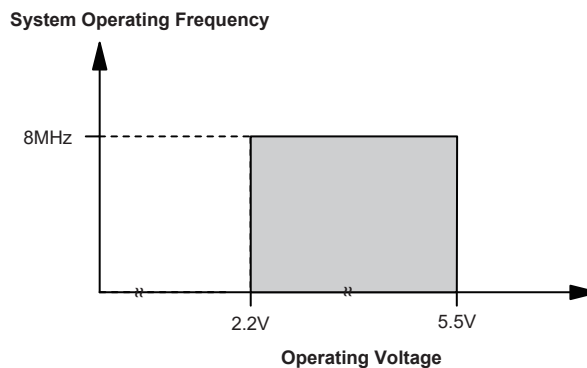
Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

Low Speed Internal Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Temp.				
f _{LIRC}	LIRC Frequency	2.2V~5.5V	25°C	-5%	32	+5%	kHz
			-40°C~85°C	-10%	32	+10%	
t _{START}	LIRC Start Up Time	—	-40°C~85°C	—	—	100	μs

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

T_a=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
t _{SST}	System Start-up Time	—	f _{SYS} =f _H ~ f _H /64, f _H =f _{HIRC}	—	16	—	t _{HIRC}
	Wake-up from Condition where f _{SYS} is Off	—	f _{SYS} =f _{SUB} =f _{LIRC}	—	2	—	t _{LIRC}
	System Start-up Time	—	f _{SYS} =f _H ~ f _H /64, f _H =f _{HIRC}	—	2	—	t _H
	Wake-up from Condition where f _{SYS} is On	—	f _{SYS} =f _{SUB} =f _{LIRC}	—	2	—	t _{SUB}
	System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode	—	f _{HIRC} switches from off → on	—	16	—	t _{HIRC}
t _{RSTD}	System Reset Delay Time Reset Source from Power-on Reset or LVR Hardware Reset	—	RR _{POR} =5V/ms	42	48	54	ms
	System Reset Delay Time LVRC/WDTC/RSTC Software Reset	—	—				
	System Reset Delay Time Reset Source from WDT Overflow	—	—	14	16	18	ms
t _{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

- Note: 1. For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.
2. The time units, shown by the symbols t_{HIRC}, are the inverse of the corresponding frequency values as provided in the frequency tables. For example t_{HIRC}=1/f_{HIRC}, t_{SYS}=1/f_{SYS} etc.
3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.
4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

$T_a = -40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports	5V	—	3.5	—	5.0	V
		—		$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Ports	3V	$V_{OL} = 0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Ports	3V	$V_{OH} = 0.9V_{DD}$, SLEDCn[m+1, m]=00B (n=0, 1; m=0 or 2 or 4 or 6)	-0.7	-1.5	—	mA
		5V		-1.5	-2.9	—	
		3V	$V_{OH} = 0.9V_{DD}$, SLEDCn[m+1, m]=01B (n=0, 1; m=0 or 2 or 4 or 6)	-1.3	-2.5	—	
		5V		-2.5	-5.1	—	
		3V	$V_{OH} = 0.9V_{DD}$, SLEDCn[m+1, m]=10B (n=0, 1; m=0 or 2 or 4 or 6)	-1.8	-3.6	—	
		5V		-3.6	-7.3	—	
		3V	$V_{OH} = 0.9V_{DD}$, SLEDCn[m+1, m]=11B (n=0, 1; m=0 or 2 or 4 or 6)	-4	-8	—	
		5V		-8	-16	—	
R_{PH}	Pull-High Resistance for I/O Ports ^(Note)	3V	—	20	60	100	k Ω
		5V		10	30	50	
I_{LEAK}	Input Leakage Current	3V	$V_{IN} = V_{DD}$ or $V_{IN} = V_{SS}$	—	—	± 1	μA
		5V		—	—	± 1	
t_{TCK}	CTCKn Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs

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

Memory Electrical Characteristics

$T_a = -40^{\circ}\text{C} \sim 85^{\circ}\text{C}$, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{RW}	V_{DD} for Read / Write	—	—	V_{DDmin}	—	V_{DDmax}	V
Flash Program Memory / Data EEPROM Memory							
t_{DEW}	Erase / Write Cycle Time – Flash Program Memory	—	—	—	2	3	ms
	Write Cycle Time – Data EEPROM Memory	—	—	—	4	6	
E_P	Cell Endurance – Flash Program Memory	—	—	10K	—	—	E/W
	Cell Endurance – Data EEPROM Memory	—	—	100K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	$T_a = 25^{\circ}\text{C}$	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

LVD/LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	-5%	2.1	+5%	V
			LVR enable, voltage select 2.55V		2.55		
			LVR enable, voltage select 3.15V		3.15		
			LVR enable, voltage select 3.8V		3.8		
V _{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 1.04V	-10%	1.04	+10%	V
			LVD enable, voltage select 2.2V	-5%	2.2	+5%	
			LVD enable, voltage select 2.4V		2.4		
			LVD enable, voltage select 2.7V		2.7		
			LVD enable, voltage select 3.0V		3.0		
			LVD enable, voltage select 3.3V		3.3		
			LVD enable, voltage select 3.6V		3.6		
			LVD enable, voltage select 4.0V		4.0		
I _{LVR/LVDBG}	Operating Current	3V	LVD enable, LVR enable, VBGEN=0	—	—	18	μA
		5V		—	20	25	
		3V	LVD enable, LVR enable, VBGEN=1	—	—	150	
		5V		—	180	200	
t _{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	18	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

Analog Front End Circuit Characteristics

24-bit A/D Converter Electrical Characteristics

V_{DD}=V_{IN}, Ta=25°C, unless otherwise specified

LDO & VCM test conditions: MCU enters SLEEP mode, other functions disabled

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{IN}	LDO Input Voltage	—	—	2.6	—	5.5	V
I _Q	LDO Quiescent Current (Including VCM Buffer)	—	LDOVS[1:0]=00B, V _{IN} =3.6V, No load	—	600	720	μA
V _{OUT_LDO}	LDO Output Voltage	—	LDOVS[1:0]=00B, V _{IN} =3.6V, I _{LOAD} =0.1mA	-5%	2.4	+5%	V
		—	LDOVS[1:0]=01B, V _{IN} =3.6V, I _{LOAD} =0.1mA		2.6		
		—	LDOVS[1:0]=10B, V _{IN} =3.6V, I _{LOAD} =0.1mA		2.9		
		—	LDOVS[1:0]=11B, V _{IN} =3.6V, I _{LOAD} =0.1mA		3.3		
ΔV _{LOAD}	LDO Load Regulation ⁽¹⁾	—	LDOVS[1:0]=00B, V _{IN} =V _{OUT_LDO} + 0.2V, 0mA ≤ I _{LOAD} ≤ 10mA	—	0.105	0.210	%/ mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{DROP_LDO}	LDO Dropout Voltage ⁽²⁾	—	LDOVS[1:0]=00B, V _{IN} =3.6V, I _{LOAD} =10mA, ΔV _{OUT_LDO} =2%	—	—	220	mV
		—	LDOVS[1:0]=01B, V _{IN} =3.6V, I _{LOAD} =10mA, ΔV _{OUT_LDO} =2%	—	—	200	mV
		—	LDOVS[1:0]=10B, V _{IN} =3.6V, I _{LOAD} =10mA, ΔV _{OUT_LDO} =2%	—	—	180	mV
		—	LDOVS[1:0]=11B, V _{IN} =3.6V, I _{LOAD} =10mA, ΔV _{OUT_LDO} =2%	—	—	160	mV
TC _{LDO}	LDO Temperature Coefficient	—	Ta=-40°C~85°C, LDOVS[1:0]=00B, V _{IN} =3.6V, I _{LOAD} =100μA	—	—	200	ppm/ °C
ΔV _{LINE_LDO}	LDO Line Regulation	—	LDOVS[1:0]=00B, 2.6V ≤ V _{IN} ≤ 5.5V, I _{LOAD} =100μA	—	—	0.7	%/V
		—	LDOVS[1:0]=00B, 2.6V ≤ V _{IN} ≤ 3.6V, I _{LOAD} =100μA	—	—	0.2	%/V
V _{OUT_VCM}	VCM Output Voltage	—	V _{IN} =3.6V, No load	-5%	1.25	+5%	V
TC _{VCM}	VCM Temperature Coefficient	—	Ta=-40°C~85°C, V _{IN} =3.6V, No load	—	—	200	ppm/ °C
ΔV _{LINE_VCM}	VCM Line Regulation	—	2.6V ≤ V _{IN} ≤ 3.6V, No load	—	—	0.3	%/V
t _{VCMS}	VCM Turn-on Stable Time	—	V _{IN} =3.6V, No load	—	—	10	ms
I _{OH_VCM}	Source Current for VCM Output Pin	—	V _{IN} =3.6V, ΔV _{OUT_VCM} =-2%	2	—	—	mA
I _{OL_VCM}	Sink Current for VCM Output Pin	—	V _{IN} =3.6V, ΔV _{OUT_VCM} =+2%	2	—	—	mA
ADC & ADC Internal Reference Voltage (Delta Sigma A/D Converter)							
V _{OREG}	Supply Voltage for ADC, PGA	—	LDOEN=0	2.4	—	3.3	V
		—	LDOEN=1	2.4	—	3.3	V
I _{ADC}	Additional Current for ADC Enable	—	—	—	400	550	μA
I _{ADSTB}	Standby Current	—	MCU enters SLEEP mode, no load	—	—	1	μA
N _R	Resolution	—	—	—	—	24	Bit
INL	Integral Non-linearity	—	V _{OREG} =3.3V, V _{REF} =1.25V, ΔSI=±450mV, PGA gain=1	—	±50	±200	ppm
NFB	Noise Free Bits	—	PGA gain=128 Data rate=10Hz	—	15.4	—	Bit
ENOB	Effective Number of Bits	—	PGA gain=128 Data rate=10Hz	—	18.1	—	Bit
f _{ADCK}	ADC Clock Frequency	—	—	40.0	409.6	440.0	kHz
f _{ADO}	ADC Output Data Rate	—	f _{MCLK} =4MHz, FLMS[2:0]=000B	4	—	521	Hz
		—	f _{MCLK} =4MHz, FLMS[2:0]=010B	10	—	1302	Hz
V _{REFP}	Reference Input Voltage	—	—	V _{REFN} +0.8	—	V _{OREG}	V
V _{REFN}		—		0	—	V _{REFP} -0.8	V
V _{REF}		—		V _{REF} =(V _{REFP} - V _{REFN})×VREFGN	0.80	—	1.75

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
PGA							
V _{CM_PGA}	Common Mode Voltage Range	—	—	0.40	—	V _{O_{REG}} -0.95	V
ΔD _I	Differential Input Voltage Range	—	Gain=PGAGN×ADGN	-V _{REF} /Gain	—	+V _{REF} /Gain	V
Temperature Sensor							
TC _{TS}	Temperature Sensor Temperature Coefficient	—	Ta=-40°C~85°C	—	175	—	μV/ °C
OPA_ADC							
I _{OPA}	Additional Current for OPA enable	—	No load	—	200	320	μA
V _{OS}	Input Offset Voltage	—	—	-2	—	+2	mV
V _{CM_OPA}	Common Mode Voltage Range	—	—	V _{SS} +0.15	—	V _{O_{REG}} -1.40	V
PSRR	Power Supply Rejection Ratio	—	—	55	90	—	dB
CMRR	Common Mode Rejection Ratio	—	—	55	90	—	dB
OPA_VCM							
I _{OPA}	Additional Current for OPA Enable	—	No load	—	200	320	μA
V _{OS}	Input Offset Voltage	—	—	-15	—	+15	mV
V _{CM_OPA}	Common Mode Voltage Range	—	—	V _{SS} +0.3	—	V _{O_{REG}} -1.4	V
PSRR	Power Supply Rejection Ratio	—	—	50	80	—	dB
CMRR	Common Mode Rejection Ratio	—	—	50	80	—	dB

- Notes: 1. Load regulation is measured at a constant junction temperature, using pulse testing with a low ON time and is guaranteed up to the maximum power dissipation. Power dissipation is determined by the input/output differential voltage and the output current. Guaranteed maximum power dissipation will not be available over the full input/output range. The maximum allowable power dissipation at any ambient temperature is $P_D = (T_{J(MAX)} - T_A) / \theta_{JA}$.
2. Dropout voltage is defined as the input voltage minus the output voltage that produces a 2% change in the output voltage from the value at appointed V_{IN}.
3. The VCM cannot be applied to applications which require sink current ability.

Effective Number of Bits (ENOB)

V_{OREG}=3.3V, V_{REF}=1.25V, FLMS[2:0]=000

Data Rate (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
5	19.7	19.8	19.6	19.7	19.7	19.6	19.2	18.6
10	19.4	19.3	19.3	19.3	19.3	19.1	18.7	18.1
20	19.0	18.8	18.7	18.9	18.8	18.6	18.2	17.5
40	18.4	18.3	18.3	18.3	18.3	18.1	17.7	17.0
80	18.1	17.9	18.0	17.9	17.9	17.6	17.2	16.5
160	17.6	17.4	17.4	17.4	17.3	17.1	16.6	15.9
320	15.8	15.8	15.9	15.8	15.9	15.9	15.8	15.3
640	14.1	14.0	14.0	14.1	14.1	14.0	14.1	14.4

$V_{OREG}=3.3V$, $V_{REF}=1.25V$, $FLMS[2:0]=010$

Data Rate (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
12.5	19.4	18.8	18.7	18.8	18.8	18.7	18.9	18.1
25	19.0	18.3	18.3	18.3	18.3	18.2	17.9	17.3
50	18.5	17.8	17.8	17.8	17.9	17.7	17.4	16.8
100	18.2	18.2	18.1	18.2	18.1	17.8	17.2	16.4
200	17.9	17.8	17.8	17.8	17.6	17.3	16.7	15.9
400	17.4	17.2	17.2	17.2	17.1	16.8	16.2	15.4
800	16.2	16.1	16.1	16.1	16.1	15.9	15.5	14.8
1600	14.5	14.5	14.5	14.4	14.5	14.5	14.3	14.0

LCD Electrical Characteristics

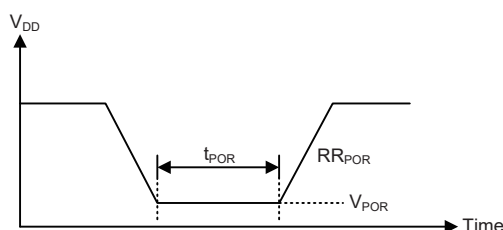
$T_a=25^{\circ}C$

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IN}	LCD Operating Voltage	—	Power supply from PLCD pin, LCDPR=0	3.0	—	5.5	V
I_{LCD}	Additional Current for LCD Enable, LCD Clock=4kHz	5V	No load, $V_A=V_{PLCD}=V_{DD}$, LCDPR=0, LCDIS[1:0]=00b	—	25.0	37.5	μA
			No load, $V_A=V_{PLCD}=V_{DD}$, LCDPR=0, LCDIS[1:0]=01b	—	50	75	
			No load, $V_A=V_{PLCD}=V_{DD}$, LCDPR=0, LCDIS[1:0]=10b	—	100	150	
			No load, $V_A=V_{PLCD}=V_{DD}$, LCDPR=0, LCDIS[1:0]=11b	—	200	300	
I_{LCDOL}	LCD COM and SEG Sink Current	3V	$V_{OL}=0.1V_{DD}$	210	420	—	μA
		5V		350	700	—	
I_{LCDOH}	LCD COM and SEG Source Current	3V	$V_{OH}=0.9V_{DD}$	-80	-160	—	μA
		5V		-180	-360	—	
V_{LCD}	PLCD Comes from Charge Pump	2.2V~ 5.5V	LCDIS[1:0]=11b, LCDPR=1, CPVS[1:0]=00b	-10%	3.3	+10%	V
		2.2V~ 5.5V	LCDIS[1:0]=11b, LCDPR=1, CPVS[1:0]=01b		3.0		
		2.2V~ 5.5V	LCDIS[1:0]=11b, LCDPR=1, CPVS[1:0]=10b		2.7		
		2.7V~ 5.5V	LCDIS[1:0]=11b, LCDPR=1, CPVS[1:0]=11b		4.5		

Power-on Reset Characteristics

Ta=-40°C~85°C

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



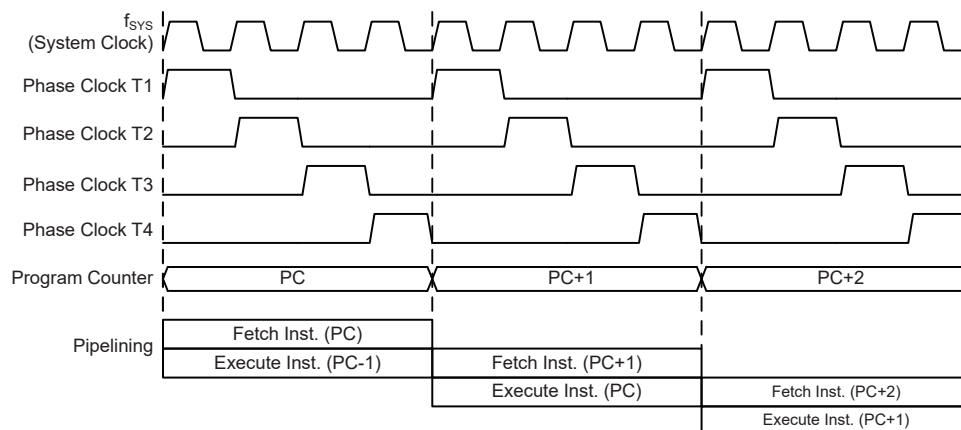
System Architecture

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

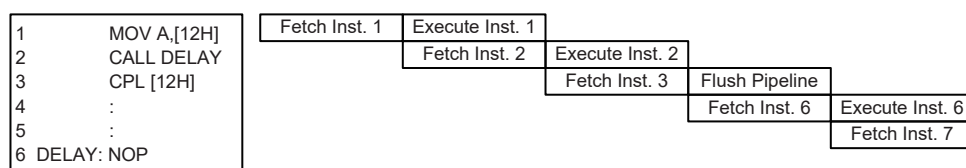
Clocking and Pipelining

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

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



System Clocking and Pipelining



Instruction Fetching

Program Counter

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

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

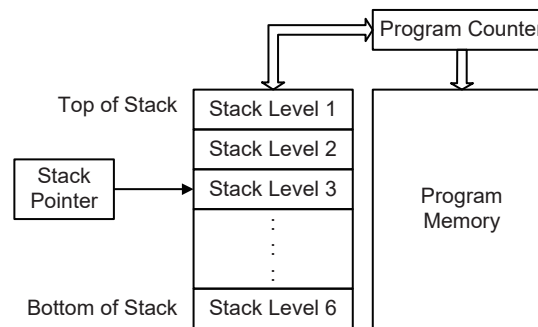
Program Counter	
Program Counter High Byte	PCL Register
PC12~PC8	PCL7~PCL0

Program Counter

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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 6 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.



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

Arithmetic and Logic Unit – ALU

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

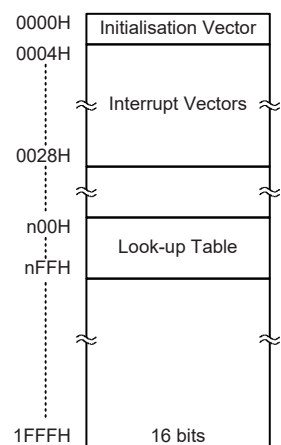
- Arithmetic operations:
 ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
 LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
 AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
 LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:
 RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
 LRR,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

Flash Program Memory

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

Structure

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



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

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

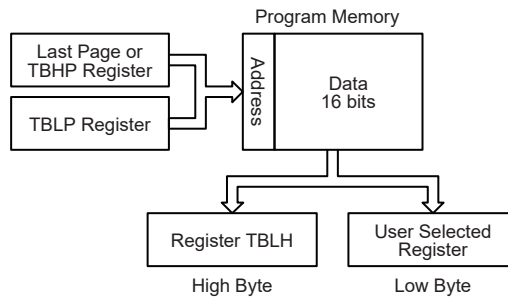


Table Program Example

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

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

Table Read Program Example

```

tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
:
mov a,06h          ; initialise low table pointer - note that this address is referenced
mov tblp,a         ; to the last page or the page that tbhp pointed
mov a,1Fh          ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer data at program
                   ; memory address "1F06H" transferred to tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrd tempreg2      ; transfers value in table referenced by table pointer data at program
                   ; memory address "1F05H" transferred to tempreg2 and TBLH in this
                   ; example the data "1AH" is transferred to tempreg1 and data "0FH" to
                   ; register tempreg2
:
:

```

```
org 1F00h          ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

In Circuit Programming – ICP

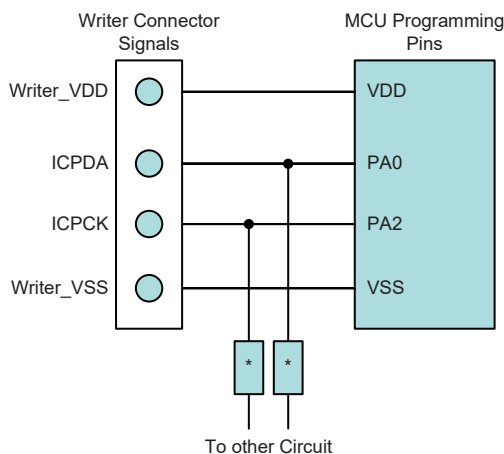
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, a means of programming the microcontroller in-circuit has provided using a 4-pin interface. This provides manufacturers 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.

The Flash MCU to Writer Programming Pin correspondence table is as follows:

Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming serial data/address
ICPCK	PA2	Programming clock
VDD	VDD	Power supply
VSS	VSS	Ground

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

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



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

On-Chip Debug Support – OCDS

An EV chip exists for the purposes of device emulation. This EV chip device also provides an “On-Chip Debug” function to debug the device during the development process. The EV chip and the actual MCU device are almost functionally compatible except for the “On-Chip Debug” function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCCK pins to the HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCCK pins in the actual 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 a more detailed OCDS description, refer to the corresponding document named “Holtek e-Link for 8-bit MCU OCDS User’s Guide”.

e-Link Pins	EV Chip Pins	Pin Description
OCDSDA	OCDSDA	On-chip debug support data/address input/output
OCDSCCK	OCDSCCK	On-chip debug support clock input
VDD	VDD	Power supply
VSS	VSS	Ground

RAM Data Memory

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

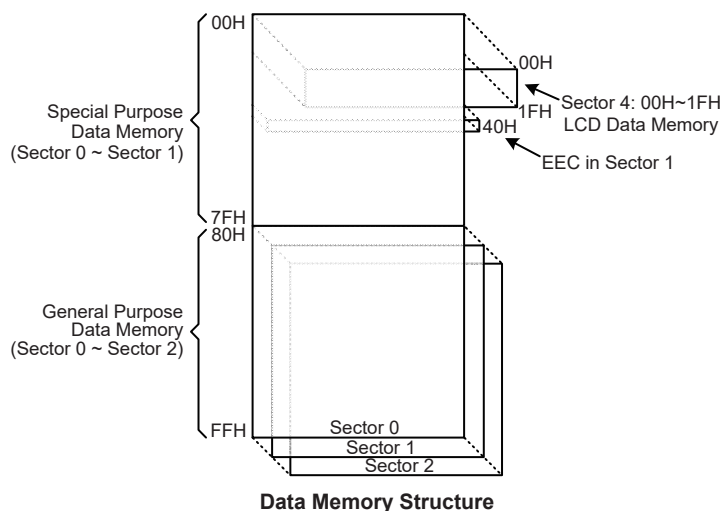
Categorized into two types, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some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

There is another area 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.

Structure

The Data Memory is subdivided into several sectors. The Special Purpose Data Memory registers are accessible in Sector 0, with the exception of the EEC register at address 40H, which are only accessible in Sector 1. Switching between the different Data Memory sectors is achieved by setting the Memory Pointers to the correct value. The start address of the Data Memory for the device is the address 00H.

Special Purpose Data Memory	LCD Data Memory		General Purpose Data Memory	
Available Sectors	Capacity	Sector: Address	Capacity	Sector: Address
0, 1	32×8	4: 00H~1FH	384×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH



Data Memory Addressing

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

Direct Addressing can be used in all sectors using the extended instructions which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except Sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address “m” in the extended instructions has 11 valid bits for this device, the high byte indicates a sector and the low byte indicates a specific address.


General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0	Sector 1		Sector 0	Sector 1
00H	IAR0		40H		EEC
01H	MP0		41H	EEA	
02H	IAR1		42H	EED	
03H	MP1L		43H		
04H	MP1H		44H		
05H	ACC		45H		
06H	PCL		46H		
07H	TBLP		47H		
08H	TBLH		48H		
09H	TBHP		49H		
0AH	STATUS		4AH		
0BH			4BH		
0CH	IAR2		4CH		
0DH	MP2L		4DH		
0EH	MP2H		4EH		
0FH	RSTFC		4FH		
10H	SCC		50H	CTM0C0	
11H	HIRCC		51H	CTM0C1	
12H			52H	CTM0DL	
13H			53H	CTM0DH	
14H	PA		54H	CTM0AL	
15H	PAC		55H	CTM0AH	
16H	PAPU		56H	CTM1C0	
17H	PAWU		57H	CTM1C1	
18H	RSTC		58H	CTM1DL	
19H	LVRC		59H	CTM1DH	
1AH	LVDC		5AH	CTM1AL	
1BH	MF10		5BH	CTM1AH	
1CH	MF11		5CH	IFS	
1DH	MF12		5DH		
1EH			5EH		
1FH	WDTC		5FH		
20H	INTEG		60H		
21H	INTC0		61H		
22H	INTC1		62H		
23H	INTC2		63H		
24H	PB		64H		
25H	PBC		65H	ADCS	
26H	PBPU		66H	ADCR0	
27H	PC		67H	ADCR1	
28H	PCC		68H	PWRC	
29H	PCPU		69H	PGAC0	
2AH	PSCR		6AH	PGAC1	
2BH	TB0C		6BH	PGACS	
2CH	TB1C		6CH	ADRL	
2DH	PAS0		6DH	ADRM	
2EH	PAS1		6EH	ADRH	
2FH	PBS0		6FH		
30H	PBS1		70H	DSOPC	
31H	PCS0		71H	DSVCMC	
32H	SPIC0		72H		
33H	SPIC1		73H	SLEDC0	
34H	SPID		74H	SLEDC1	
35H	USR		75H		
36H	UCR1		76H		
37H	UCR2		77H		
38H	TXR_RXR		78H	LCDC0	
39H	BRG		79H	LCDCP	
3AH			7AH	COMS	
3BH			7BH		
3CH			7CH		
3DH			7DH		
3EH			7EH		
3FH			7FH		

 : Unused, read as 00H

 : Reserved, cannot be changed

Special Purpose Data Memory

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, IAR2

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

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

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

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

Indirect Addressing Program Example 1

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

Indirect Addressing Program Example 2

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

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

Direct Addressing Program Example using extended instructions

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

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

Accumulator – ACC

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

Program Counter Low Register – PCL

To provide additional program control functions, the low byte of the Program Counter is made accessible to programmers by locating it within the Special Purpose area of the Data Memory. By manipulating this register, direct jumps to other program locations are easily implemented. Loading a value directly into this PCL register will cause a jump to the specified Program Memory location, however, as the register is only 8-bit wide, only jumps within the current Program Memory page are permitted. When such operations are used, note that a dummy cycle will be inserted.

Look-up Table Registers – TBLP, TBHP, TBLH

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP and TBHP are the table pointers and indicate the location where the table data is located. Their value must be setup before any table read commands are executed. Their value can be changed, for example using the “INC” or “DEC” instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

This 8-bit register contains the SC flag, CZ flag, zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

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

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

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

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

• **STATUS Register**

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

“x”: Unknown

- Bit 7 **SC**: The result of the “XOR” operation which is performed by the OV flag and the MSB of the instruction operation result.
- Bit 6 **CZ**: The operational result of different flags for different instructions.
For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.
For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the “AND” operation result which is performed by the previous operation CZ flag and current operation zero flag.
For other instructions, the CZ flag will not be affected.
- Bit 5 **TO**: Watchdog Time-Out Flag
0: After power up or executing the “CLR WDT” or “HALT” instruction
1: A watchdog time-out occurred.
- Bit 4 **PDF**: Power Down Flag
0: After power up or executing the “CLR WDT” instruction
1: By executing the “HALT” instruction
- Bit 3 **OV**: Overflow Flag
0: No overflow
1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa.
- Bit 2 **Z**: Zero Flag
0: The result of an arithmetic or logical operation is not zero
1: The result of an arithmetic or logical operation is zero
- Bit 1 **AC**: Auxiliary flag
0: No auxiliary carry
1: An operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction
- Bit 0 **C**: Carry Flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation
The “C” flag is also affected by a rotate through carry instruction.

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 128×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and a data register in Sector 0 and a single control register in Sector 1.

EEPROM Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEA	—	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM Register List

• EEA Register

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

Bit 7 Unimplemented, read as “0”

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

• EED Register

Bit	7	6	5	4	3	2	1	0
Name	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **EED7~EED0**: Data EEPROM Data bit 7 ~ bit 0

• **EEC Register**

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

Bit 7~4 Unimplemented, read as “0”

Bit 3 **WREN**: Data EEPROM Write Enable

0: Disable

1: Enable

This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2 **WR**: EEPROM Write Control

0: Write cycle has finished

1: Activate a write cycle

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

Bit 1 **RDEN**: Data EEPROM Read Enable

0: Disable

1: Enable

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

Bit 0 **RD**: EEPROM Read Control

0: Read cycle has finished

1: Activate a read cycle

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

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

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

3. Ensure that the write operation is totally complete before changing the EEC register content.

Reading Data from the EEPROM

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

Writing Data to the EEPROM

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

Write Protection

Protection against inadvertent write operation is provided in several ways. After the device is powered-on the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on the Memory Pointer high byte register, MP1H or MP2H, will be reset to zero, which means that Data Memory Sector 0 will be selected. As the EEPROM control register is located in Sector 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

EEPROM Interrupt

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

Programming Considerations

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

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

Programming Examples

Reading data from the EEPROM – polling method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, 40H                ; setup memory pointer MP1L
MOV MP1L, A               ; MP1L points to EEC register
MOV A, 01H                ; setup memory pointer MP1H
MOV MP1H, A
SET IAR1.1                ; set RDEN bit, enable read operations
SET IAR1.0                ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                 ; check for read cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read if no more read operations are required
CLR MP1H
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

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

Writing Data to the EEPROM – polling method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
MOV A, 040H               ; setup memory pointer MP1L
MOV MP1L, A               ; MP1L points to EEC register
MOV A, 01H                ; setup memory pointer MP1H
MOV MP1H, A
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit - executed immediately
                        ; after set WREN bit

SET EMI
BACK:
SZ IAR1.2                 ; check for write cycle end
JMP BACK
CLR MP1H
```

Oscillators

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

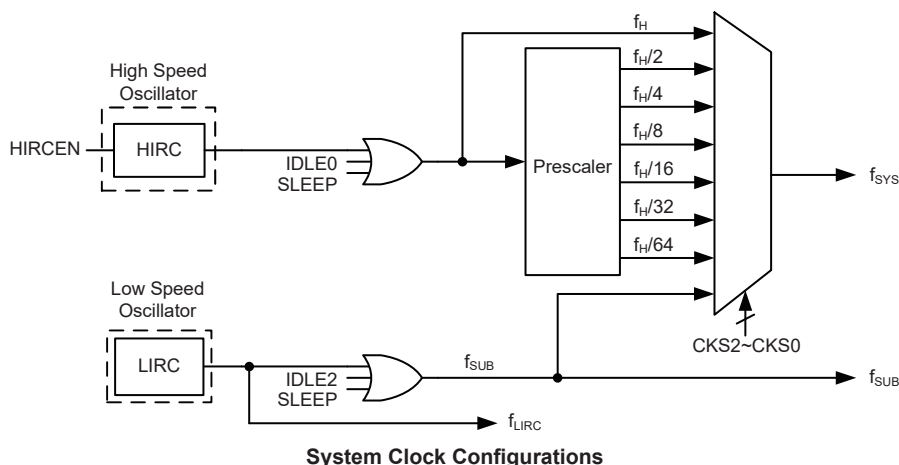
Oscillator Overview

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. Fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options are selected through the registers. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. With the capability of dynamically switching between fast and slow system clock, the device have the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

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

System Clock Configurations

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



Internal RC Oscillator – HIRC

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

Internal 32kHz Oscillator – LIRC

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

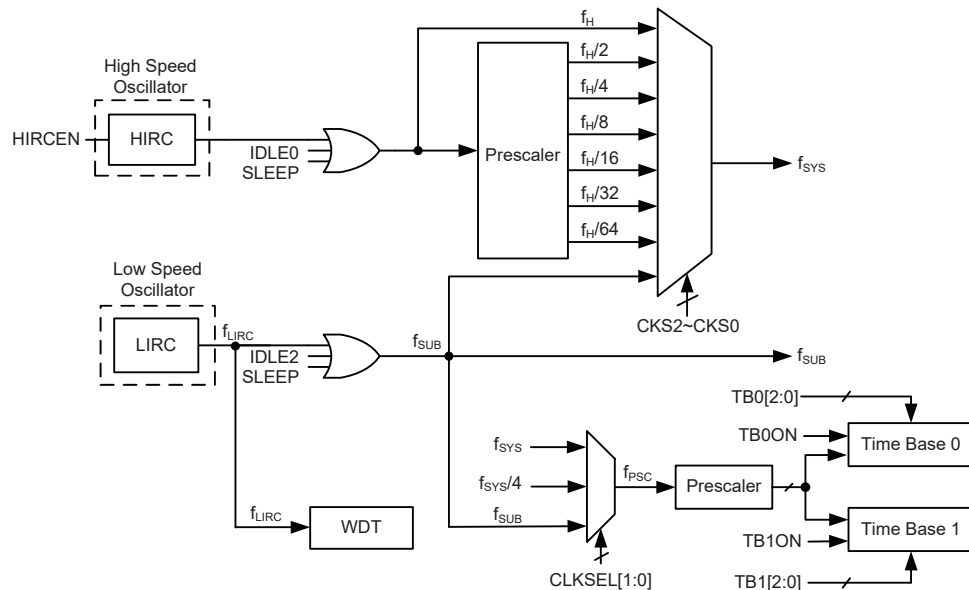
Operating Modes and System Clocks

Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice versa, lower speed clocks reduce current consumption. As both high and low speed clock sources are provided 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 different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock selections using register programming, a clock system can be configured to obtain maximum application performance.

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



Device Clock Configurations

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

System Operation Modes

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

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

"x": Don't care

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

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

FAST Mode

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

SLOW Mode

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

SLEEP Mode

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

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

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

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

System Operating Mode Control Register List

• SCC Register

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

Bit 7~5 **CKS2~CKS0: System Clock Selection**

000: f_H
 001: $f_H/2$
 010: $f_H/4$
 011: $f_H/8$
 100: $f_H/16$
 101: $f_H/32$
 110: $f_H/64$
 111: f_{SUB}

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

Bit 4~2 Unimplemented, read as “0”

Bit 1 **FHIDEN: High Frequency Oscillator Control when CPU is Switched Off**

0: Disable
 1: Enable

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

Bit 0 **FSIDEN: Low Frequency Oscillator Control when CPU is Switched Off**

0: Disable
 1: Enable

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

• **HIRCC Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1 **HIRCF**: HIRC Oscillator Stable Flag
 0: HIRC unstable
 1: HIRC stable

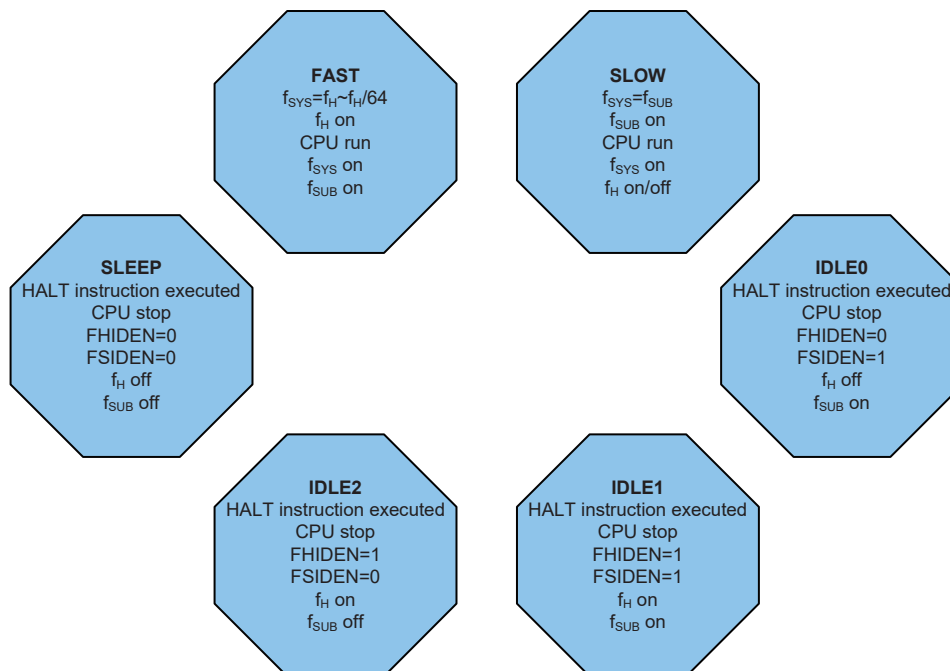
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

Bit 0 **HIRCEN**: HIRC Oscillator Enable Control
 0: Disable
 1: Enable

Operating Mode Switching

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

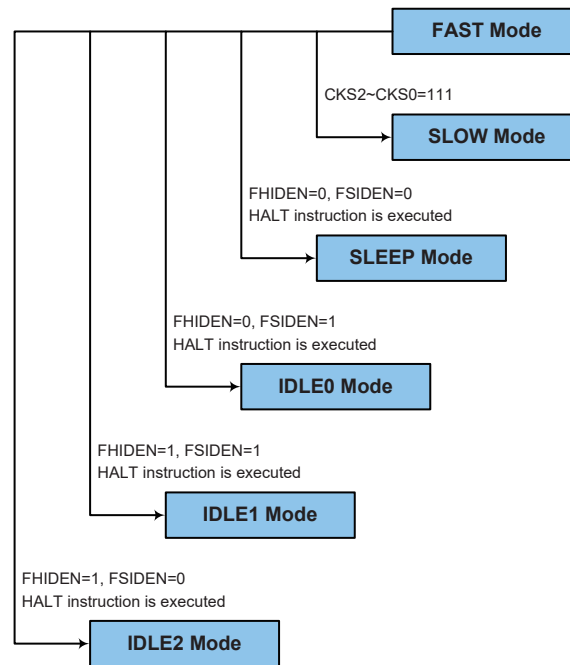
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When an HALT instruction is executed, whether the device enters the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



FAST Mode to SLOW Mode Switching

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

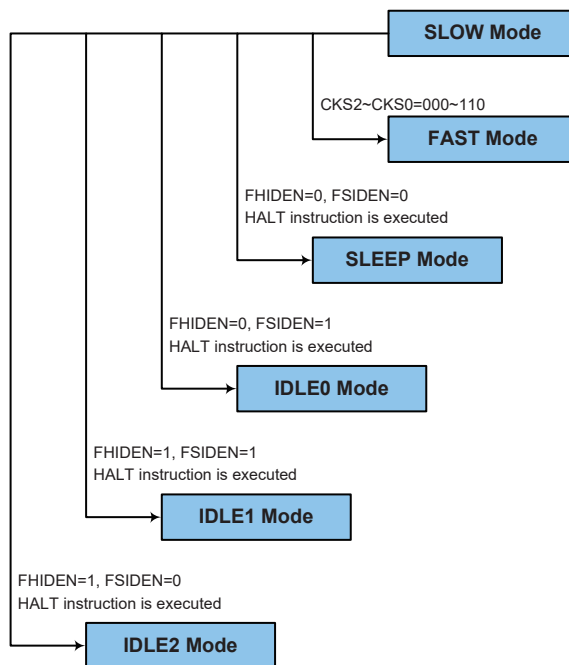
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

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

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



Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

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

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

Entering the IDLE2 Mode

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

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

Standby Current Considerations

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

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

Wake-up

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

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

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

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

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{LIRC} , which is in turn supplied by the LIRC oscillator. 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. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with V_{DD} , temperature and process variations.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable and reset MCU operation.

• WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT Function Software Control

10101: Disable
01010: Enable
Others: Reset MCU

When these bits are changed by the environmental noise or software setting to reset the microcontroller, the reset operation will be activated after a delay time, t_{SRESET} , and the WRF bit in the RSTFC register will be set high.

Bit 2~0 **WS2~WS0**: WDT Time-out Period Selection

000: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$

These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period.

• RSTFC Register

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

“x”: Unknown

Bit 7~4 Unimplemented, read as “0”

Bit 3 **RSTF**: Reset Control Register Software Reset Flag

Refer to the “Internal Reset Control” section

Bit 2 **LVRF**: LVR Function Reset Flag

Refer to the “Low Voltage Reset” section

- Bit 1 **LRF**: LVR Control Register Software Reset Flag
Refer to the “Low Voltage Reset” section
- Bit 0 **WRF**: WDT Control Register Software Reset Flag
0: Not occur
1: Occurred
- This bit is set high by the WDT Control register software reset and cleared by the application program. Note that this bit can only be cleared to zero by the application program.

Watchdog Timer Operation

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

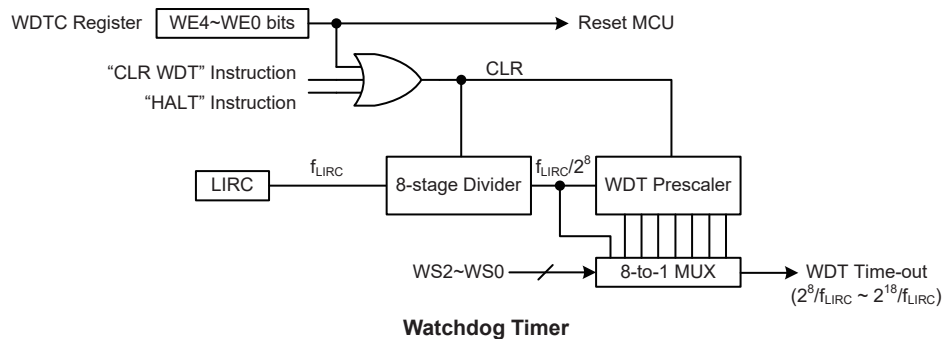
WE4 ~ WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other values	Reset MCU

Watchdog Timer Enable/Disable Control

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

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

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



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.

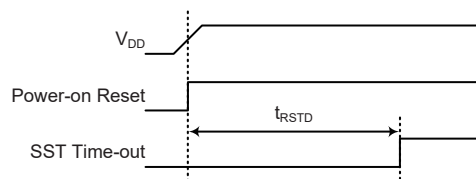
Another type of reset is when the Watchdog Timer overflows and resets. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, is implemented in situations where the power supply voltage falls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

Reset Functions

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

Power-on Reset

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



Power-On Reset Timing Chart

Internal Reset Control

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

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

Internal Reset Function Control

• **RSTC Register**

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

Bit 7~0 **RSTC7~RSTC0**: Reset Function Control

01010101: No operation

10101010: No operation

Other values: Reset MCU

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

• **RSTFC Register**

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

“x”: Unknown

Bit 7~4 Unimplemented, read as “0”

Bit 3 **RSTF**: Reset Control Register Software Reset Flag

0: Not occurred

1: Occurred

This bit is set to 1 by the RSTC control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

Bit 2 **LVRF**: LVR Function Reset Flag

Refer to the “Low Voltage Reset” section

Bit 1 **LRF**: LVR Control Register Software Reset Flag

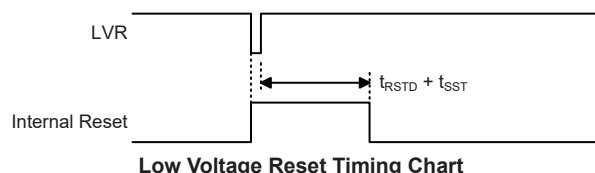
Refer to the “Low Voltage Reset” section

Bit 0 **WRF**: WDT Control Register Software Reset Flag

Refer to the “Watchdog Timer Control Register” section

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled in the FAST and SLOW modes with a specific LVR voltage V_{LVR} . If the supply voltage of the device drops to within a range of $0.9V \sim V_{LVR}$ such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between $0.9V \sim V_{LVR}$ must exist for a time greater than that specified by t_{LVR} in the LVD/LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise or software setting, the LVR will reset the device after a delay time, t_{SRESET} . When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the IDLE/SLEEP mode.



• **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: MCU reset (register is reset to POR value)

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

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

• **RSTFC Register**

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

“x”: Unknown

Bit 7~4 Unimplemented, read as “0”

Bit 3 **RSTF**: Reset Control Register Software Reset Flag

Refer to the “Internal Reset Control” section

Bit 2 **LVRF**: LVR Function Reset Flag

0: Not occur

1: Occurred

This bit is set high when a specific Low Voltage Reset situation occurs. This bit can only be cleared to zero by the application program.

Bit 1 **LRF**: LVR Control Register Software Reset Flag

0: Not occur

1: Occurred

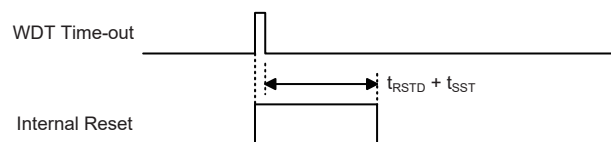
This bit is set high if the LVRC register contains any non-defined LVR voltage register values. This in effect acts like a software-reset function. This bit can only be cleared to zero by the application program.

Bit 0 **WRF**: WDT Control register software reset flag

Refer to the “Watchdog Timer Control Register” section

Watchdog Time-out Reset during Normal Operation

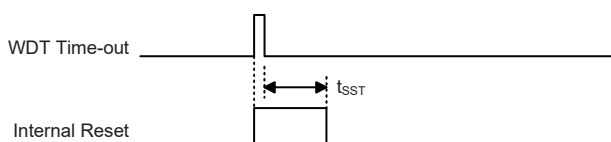
When a Watchdog time-out Reset occurs during normal operation, the Watchdog time-out flag TO will be set to “1”.



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 zero and the TO flag will be set high. Refer to the System Start Up Time Characteristics for t_{SST} details.



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

Reset Initial Conditions

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

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

“u”: Unchanged

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

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

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

Register Name	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---x xxxx	---u uuuu	---u uuuu	---u uuuu
STATUS	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
IAR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- 0x00	---- u1uu	---- uuuu	---- uuuu
SCC	000- --00	000- --00	000- --00	uuu- --uu
HIRCC	---- --01	---- --01	---- --01	---- --uu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTC	0101 0101	0101 0101	0101 0101	uuuu uuuu
LVRC	0101 0101	uuuu uuuu	0101 0101	uuuu uuuu
LVDC	--00 0000	--00 0000	--00 0000	--uu uuuu
MF10	--00 --00	--00 --00	--00 --00	--uu --uu
MF11	--00 --00	--00 --00	--00 --00	--uu --uu
MF12	--00 --00	--00 --00	--00 --00	--uu --uu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
INTEG	---- 0000	---- 0000	---- 0000	---- uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	-000 -000	-000 -000	-000 -000	-uuu -uuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	---- ---1	---- ---1	---- ---1	---- ---u
PCC	---- ---1	---- ---1	---- ---1	---- ---u
PCPU	---- ---0	---- ---0	---- ---0	---- ---u
PSCR	---- --00	---- --00	---- --00	---- --uu
TB0C	0--- -000	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	0--- -000	u--- -uuu
PAS0	00-- 00--	00-- 00--	00-- 00--	uu-- uu--
PAS1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 ----	0000 ----	0000 ----	uuuu ----
PBS1	--00 0000	--00 0000	--00 0000	--uu uuuu
PCS0	---- --00	---- --00	---- --00	---- --uu
SPIC0	111- --00	111- --00	111- --00	uuu- --uu

Register Name	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
SPIC1	--00 0000	--00 0000	--00 0000	--uu uuuu
SPID	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
USR	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
EEA	-000 0000	-000 0000	-000 0000	-uuu uuuu
EED	0000 0000	0000 0000	0000 0000	uuuu uuuu
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
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
IFS	---- --00	---- --00	---- --00	---- --uu
ADCS	---0 0000	---0 0000	---0 0000	---u uuuu
ADCR0	0010 00-0	0010 00-0	0010 00-0	uuuu uu-u
ADCR1	000- -00-	000- -00-	000- -00-	uuu- -uu-
PWRC	0--- -000	0--- -000	0--- -000	u--- -uuu
PGAC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
PGAC1	-000 000-	-000 000-	-000 000-	-uuu uuu-
PGACS	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADRL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRM	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
DSOPC	0--- 0000	0--- 0000	0--- 0000	u--- uuuu
DSVCMC	---- 1010	---- 1010	---- 1010	---- uuuu
SLEDC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC1	---- --00	---- --00	---- --00	---- --uu
LCDC0	0--- -000	0--- -000	0--- -000	u--- -uuu
LCDCP	---0 0-00	---0 0-00	---0 0-00	---u u-uu
COMS	---- --00	---- --00	---- --00	---- --uu
EEC	---- 0000	---- 0000	---- 0000	---- uuuu

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

Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA~PC. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction “MOV A, [m]”, where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

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

“—”: Unimplemented

Input/Output Logic Function Register List

Pull-high Resistors

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

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

• PxPU Register

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

PxPUn: I/O Port x Pin Pull-high Function Control

0: Disable

1: Enable

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

Port A Wake-up

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

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

• PAWU Register

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

Bit 7~0 **PAWU7~PAWU0**: Port A Pin Wake-up Control

0: Disable

1: Enable

I/O Port Control Registers

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

• PxC Register

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

PxCn: I/O Port x Pin Type Selection

0: Output

1: Input

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

I/O Port Source Current Control

The device supports different source current driving capability for each I/O port. With the corresponding selection registers, SLEDC0 and SLEDC1, each I/O port can support four levels of the source current driving capability. Users should refer to the Input/Output Characteristics section to select the desired source current for different applications.

• **SLEDC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **SLEDC07~SLEDC06:** PB7~PB4 Source Current Selection

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

Bit 5~4 **SLEDC05~SLEDC04:** PB3~PB0 Source Current Selection

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

Bit 3~2 **SLEDC03~SLEDC02:** PA7~PA4 Source Current Selection

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

Bit 1~0 **SLEDC01~SLEDC00:** PA3~PA0 Source Current Selection

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

• **SLEDC1 Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **SLEDC11~SLEDC10:** PC0 Source Current Selection

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

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, labeled as IFS, which can select

the desired functions of the multi-function pin-shared pins. In addition, the device also contains a COMS register, which can be used to determine the actual pin function when the LCD SEG and COM functions share the same pin.

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	—	—	PAS03	PAS02	—	—
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	—	—	—	—
PBS1	—	—	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	—	—	—	—	—	—	PCS01	PCS00
IFS	—	—	—	—	—	—	IFS1	IFS0
COMS	—	—	—	—	—	—	COMS1	COMS0

Pin-shared Function Selection Register List

• **PAS0 Register**

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

Bit 7~6 **PAS07~PAS06:** PA3 Pin-shared Function Selection
 00/11: PA3
 01: CTP1
 10: AN7

Bit 5~4 Unimplemented, read as “0”

Bit 3~2 **PAS03~PAS02:** PA1 Pin-shared Function Selection
 00/01/11: PA1/INT0
 10: AN6

Bit 1~0 Unimplemented, read as “0”

• **PAS1 Register**

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

- Bit 7~6 **PAS17~PAS16:** PA7 Pin-shared Function Selection
00: PA7
01: CTP0B
10: SPISDI
11: RX
- Bit 5~4 **PAS15~PAS14:** PA6 Pin-shared Function Selection
00/11: PA6
01: SPISDO
10: TX
- Bit 3~2 **PAS13~PAS12:** PA5 Pin-shared Function Selection
00/11: PA5
01: CTP1B
10: SPISCS
- Bit 1~0 **PAS11~PAS10:** PA4 Pin-shared Function Selection
00/10/11: PA4
01: SPISCK

• **PBS0 Register**

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

- Bit 7~6 **PBS07~PBS06:** PB3 Pin-shared Function Selection
00/10/11: PB3
01: OPIP
- Bit 5~4 **PBS05~PBS04:** PB2 Pin-shared Function Selection
00/10/11: PB2/INT1
01: LVDIN
- Bit 3~0 Unimplemented, read as “0”

• **PBS1 Register**

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

- Bit 7~6 Unimplemented, read as “0”
- Bit 5~4 **PBS15~PBS14:** PB6 Pin-shared Function Selection
00/11: PB6
01: SPISDO
10: TX
- Bit 3~2 **PBS13~PBS12:** PB5 Pin-shared Function Selection
00: PB5
01: SPISDI
10: RX
11: AN5

Bit 1~0 **PBS11~PBS10**: PB4 Pin-shared Function Selection
 00/11: PB4/CTCK0
 01: OPIN
 10: AN4

• **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PCS01	PCS00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **PCS01~PCS00**: PC0 Pin-shared Function Selection
 00/10/11: PC0
 01: CTP0

• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	IFS1	IFS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1 **IFS1**: SPISDI input source pin selection
 0: PB5
 1: PA7

Bit 0 **IFS0**: RX input source pin selection
 0: PA7
 1: PB5

• **COMS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	COMS1	COMS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

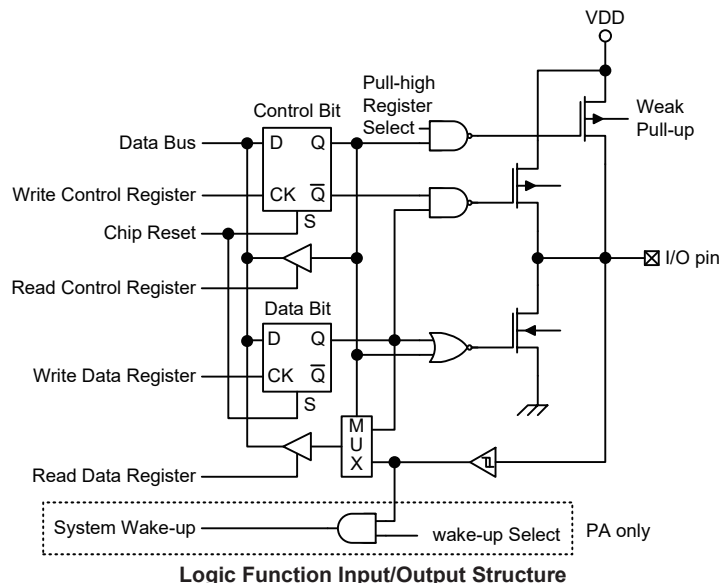
Bit 7~2 Unimplemented, read as “0”

Bit 1 **COMS1**: SEG30/COM5 Pin-shared Function Selection
 0: SEG30
 1: COM5

Bit 0 **COMS0**: SEG31/COM4 Pin-shared Function Selection
 0: SEG31
 1: COM4

I/O Pin Structures

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



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 Timer Modules, generally abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Compare Match Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two interrupts. The addition of input and output pins for TM ensures that users are provided with timing units with a wide and flexible range of features.

Introduction

The device contains two Compact Type TM units, with its individual reference name, CTM. The main features of CTM are summarised in the accompanying table.

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

TM Function Summary

TM Operation

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

TM Clock Source

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

TM Interrupts

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

TM External Pins

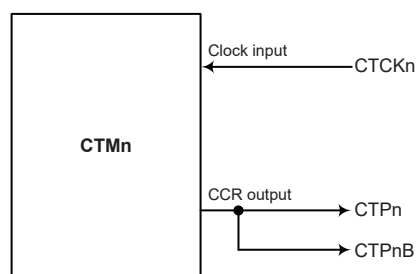
The TMs each has one input pin with the label CTCKn. The CTM input pin CTCKn, is essentially a clock source for the TM and is selected using the CTnCK2~CTnCK0 bits in the CTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The TM input pin can be chosen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

The TMs each has two output pins with the label CTPn and CTPnB. 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 CTPn and CTPnB output pins are also the pins where the TM generates the PWM output waveform.

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

CTM	
Input	Output
CTCK0 CTCK1	CTP0, CTP0B CTP1, CTP1B

TM External Pins

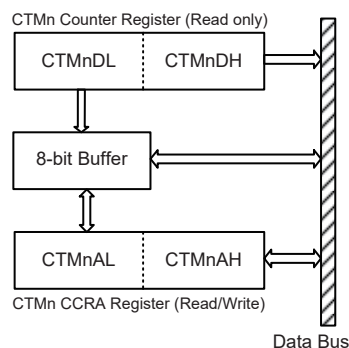


CTMn Function Pin Block Diagram (n=0~1)

Programming Considerations

The TM Counter Registers and the Compare CCRA register, has 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 register is 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 low byte registers, named CTMnAL, using the following access procedures. Accessing the CCRA low byte registers without following these access procedures will result in unpredictable values.

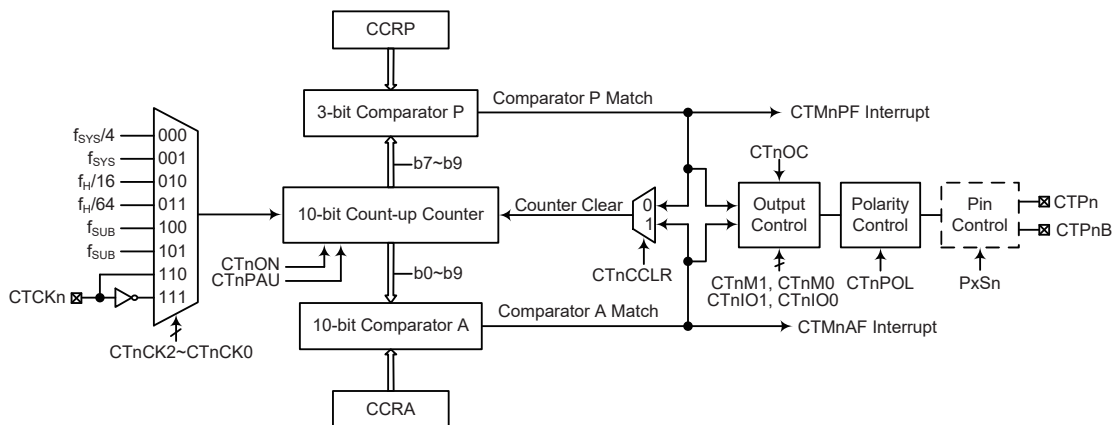


The following steps show the read and write procedures:

- Writing Data to CCRA
 - ♦ Step 1. Write data to Low Byte CTMnAL
 - Note that here data is only written to the 8-bit buffer.
 - ♦ Step 2. Write data to High Byte CTMnAH
 - Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA
 - ♦ Step 1. Read data from the High Byte CTMnDH, CTMnAH
 - Here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
 - ♦ Step 2. Read data from the Low Byte CTMnDL, CTMnAL
 - This step reads data from the 8-bit buffer.

Compact Type TM – CTM

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



Note: CTPnB is the inverse signal of CTPn.

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

Compact Type TM Operation

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

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the CTnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a CTMn interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock

sources including an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

Compact Type TM Register Description

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

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

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

• CTMnC0 Register

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

Bit 7 **CTnPAU**: CTMn Counter Pause Control

0: Run
1: Pause

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

Bit 6~4 **CTnCK2~CTnCK0**: Select CTMn Counter clock

000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: CTCKn rising edge clock
111: CTCKn falling edge clock

These three bits are used to select the clock source for the CTMn. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the 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, clearing the bit to 0 disables the CTMn. Clearing this bit to zero will stop the counter from counting and turn off the CTMn which will reduce its power consumption. When the bit changes state from low to high the internal counter value

will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.

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

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

Comparator P Match Period

- 000: 1024 CTMn clocks
- 001: 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 Output Mode
- 11: Timer/Counter Mode

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

Bit 5~4 **CTnIO1~CTnIO0**: Select CTMn 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 Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the CTnIO1 and CTnIO0 bits only after the CTMn has been switched off. Unpredictable PWM outputs will occur if the CTnIO1 and CTnIO0 bits are changed when The CTMn is running.

Bit 3 **CTnOC:** CTPn Output control bit

Compare Match Output Mode

0: Initial low

1: Initial high

PWM Output Mode

0: Active low

1: Active high

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

Bit 2 **CTnPOL:** CTPn Output polarity Control

0: Non-invert

1: Invert

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

Bit 1 **CTnDPX:** CTMn PWM period/duty Control

0: CCRP – period; CCRA – duty

1: CCRP – duty; CCRA – period

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

Bit 0 **CTnCCLR:** Select CTMn Counter clear condition

0: CTMn Comparatr P match

1: CTMn Comparatr 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 Output Mode.

• **CTMnDL Register**

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

Bit 7~0 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

Compact Type TM Operating Modes

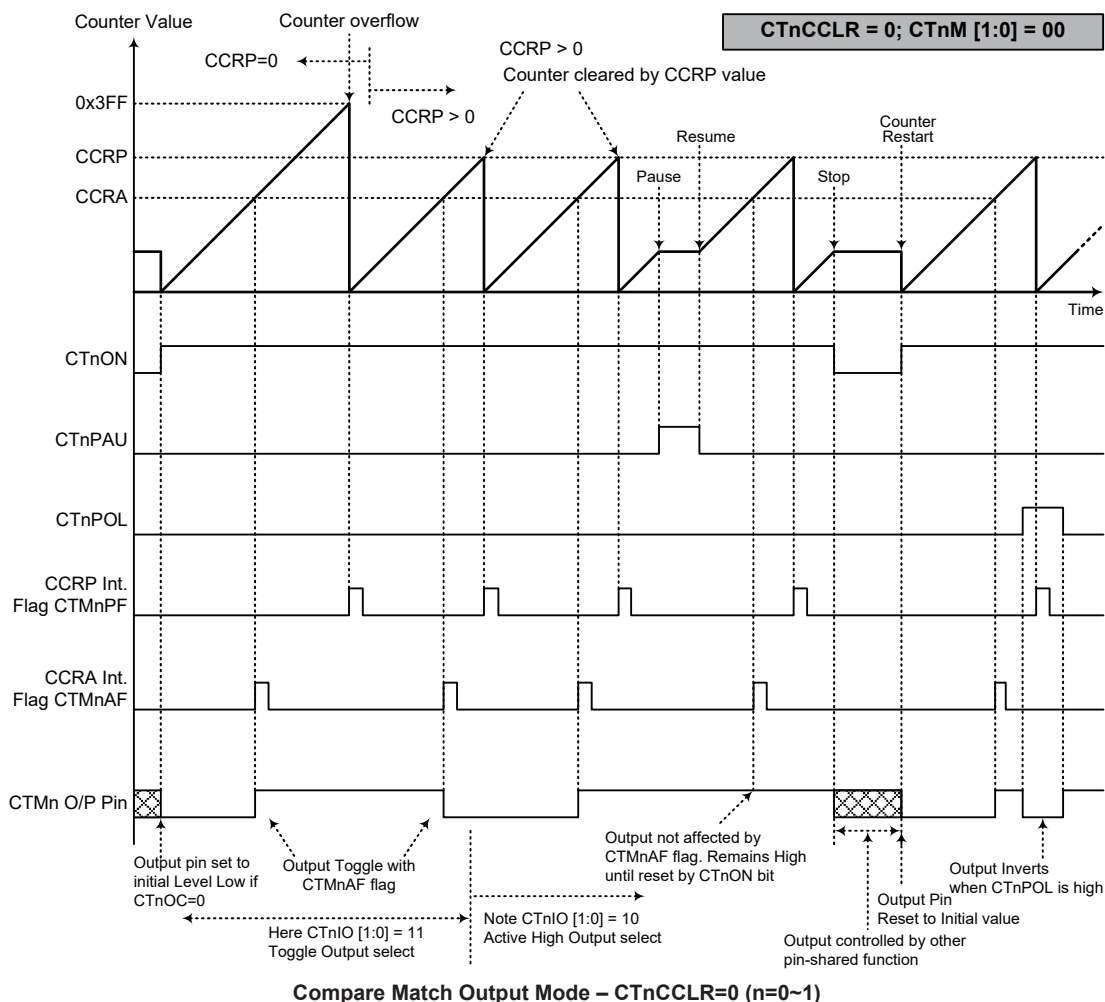
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

Compare Match Output Mode

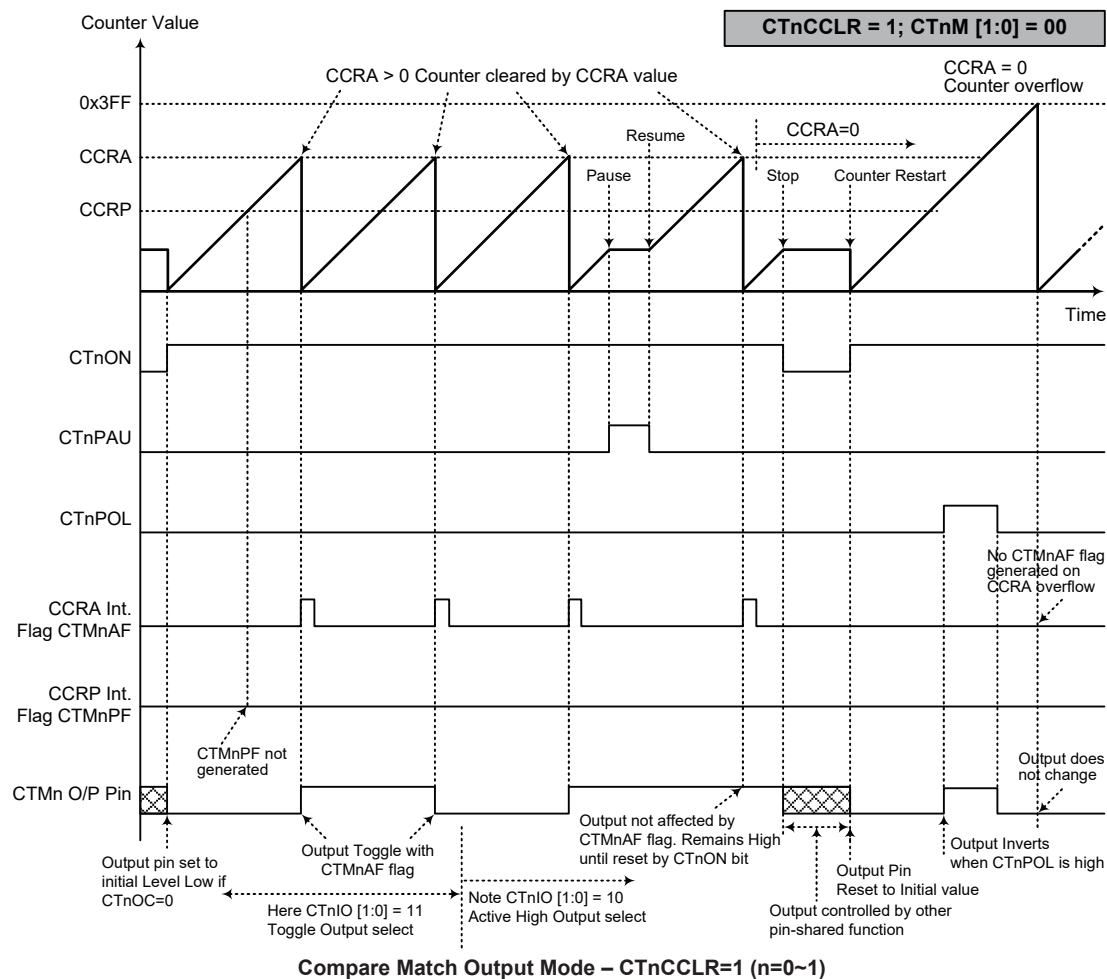
To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the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

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

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



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



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

Timer/Counter Mode

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

PWM Output Mode

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

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the CTnCCLR bit has no effect 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 CTMn output pin to a fixed high or low level. The CTnPOL bit is used to reverse the polarity of the PWM output waveform.

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

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

If $f_{SYS}=8\text{MHz}$, CTMn clock source is $f_{SYS}/4$, CCRP=2, CCRA=128,

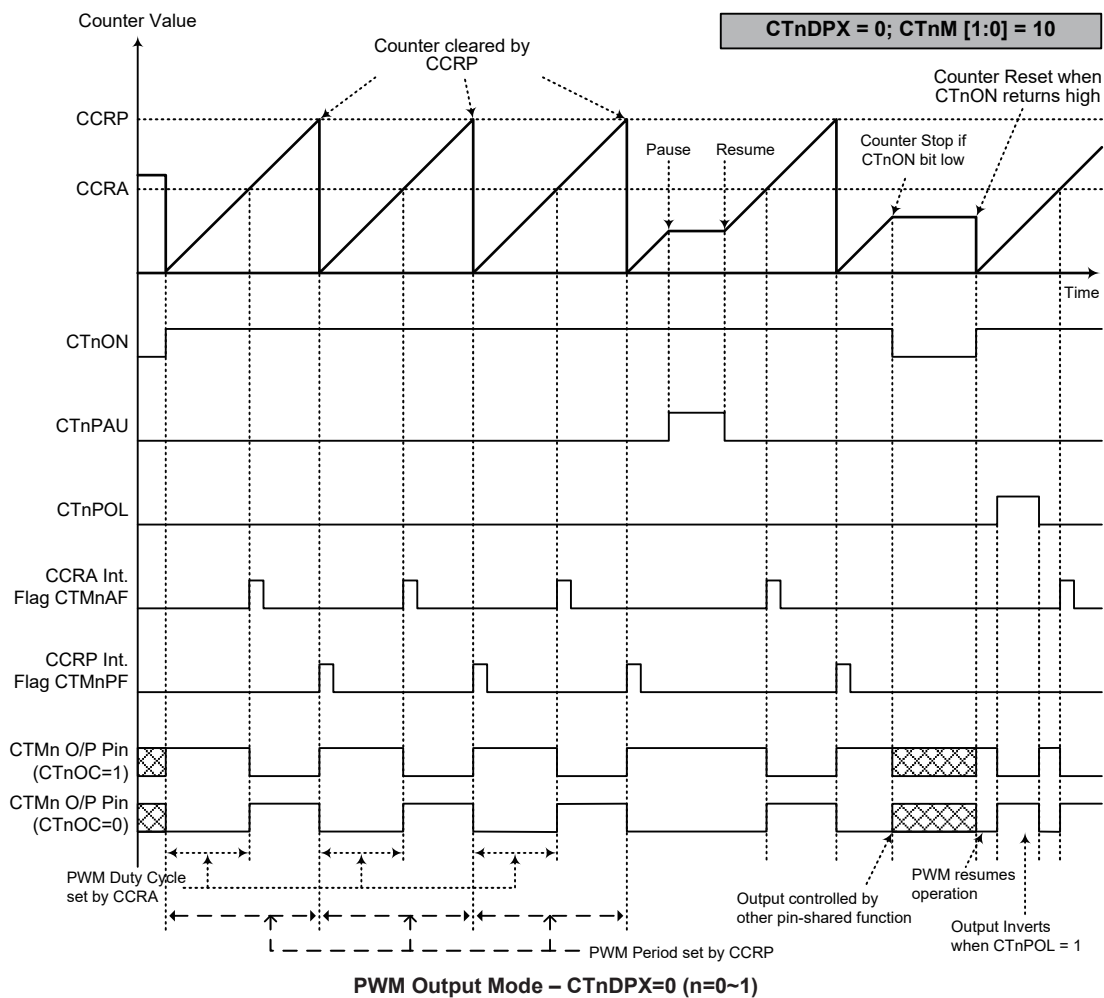
The CTMn PWM output frequency= $(f_{SYS}/4)/(2 \times 128)=f_{SYS}/1024=7.812\text{kHz}$, duty= $128/(2 \times 128)=50\%$.

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

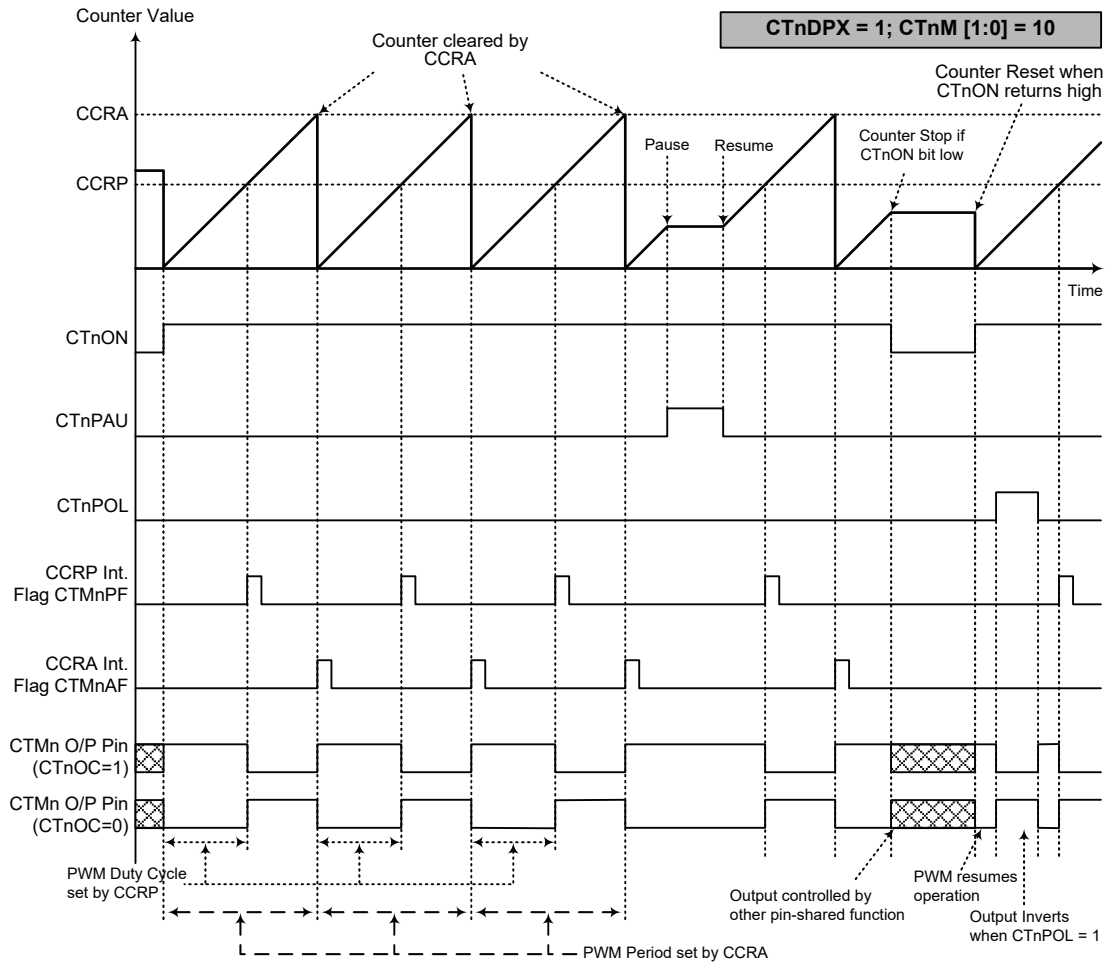
• 10-bit CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=1

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

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

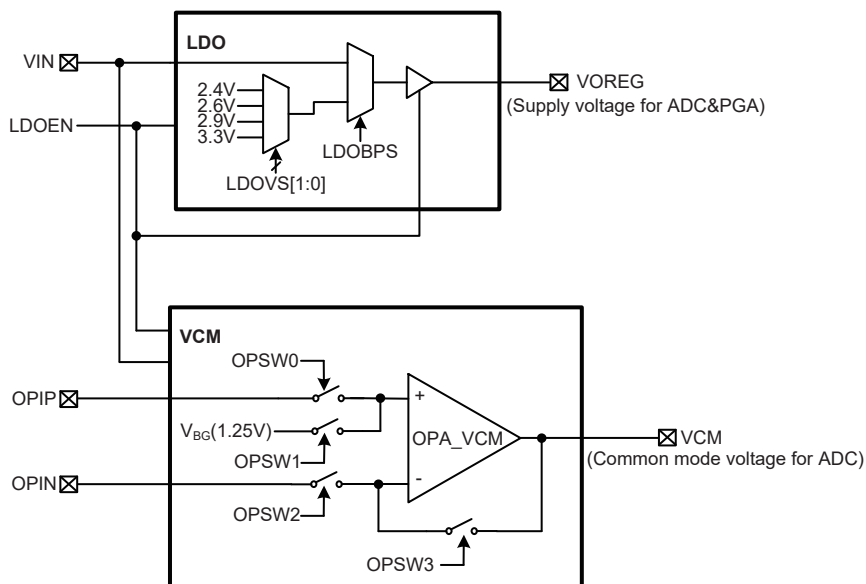


- Note: 1. Here CTnDPX=0 – Counter cleared by CCRP
2. A counter clear sets PWM Period
3. The internal PWM function continues running even when CTnIO[1:0]=00 or 01
4. The CTnCCLR bit has no influence on PWM operation



PWM Output Mode – CTnDPX=1 (n=0~1)

- Note: 1. Here CTnDPX=1 – Counter cleared by CCRA
2. A counter clear sets 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



Internal Power Supply Block Diagram

Register bits		Output Voltage		
ADOFF	LDOEN	Bandgap	VOREG	VCM
1	0	Off	Disable	Disable
1	1	On	Enable	Enable
0	0	On	Disable	Enable
0	1	On	Enable	Enable

Power Control Table

• **PWRC Register**

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

Bit 7 **LDOEN**: LDO function control bit

0: Disable

1: Enable

If the LDO is disabled, there will be no power consumption and LDO output will be in a low level by a weakly pull-low resistor.

Bit 6~3 Unimplemented, read as “0”

Bit 2 **LDOBPS**: LDO Bypass function control bit

0: Disable

1: Enable

Bit 1~0 **LDOVS1~LDOVS0**: LDO output voltage selection

00: 2.4V

01: 2.6V

10: 2.9V

11: 3.3V

• **DSVCMC Register**

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

Bit 7~4 Unimplemented, read as “0”

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

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

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

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

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

A/D Data Rate Definition

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

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

f_{ADCK} : A/D clock input, derived from f_{MCLK}/N .

f_{MCLK} : A/D clock source, derived from f_{SYS} or $f_{SYS}/2/(ADCK+1)$ using the ADCK bit field.

N: A constant divided factor that can be equal to 30 or 12 determined by the FLMS bit field.

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

OSR: Oversampling rate determined by the ADOR field.

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

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

A/D Converter Register Description

Overall operation of the A/D converter is controlled by using 12 registers. Three read only registers exist to store the A/D converter data 24-bit value. A control register PWRC is used to control the required bias and supply voltages for PGA and A/D converter, another control register DSVCMC is used to control the switches for OPA configuration in the VCM, these registers are described in the “Internal Power Supply” section. A control register named DSOPC is used to control the low noise operational amplifier. The remaining 6 registers are control registers which set up the gain selections and control functions of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PWRC	LDOEN	—	—	—	—	LDOBPS	LDOVS1	LDOVS0
PGAC0	—	VGS1	VGS0	AGS1	AGS0	PGS2	PGS1	PGS0
PGAC1	—	INIS	INX1	INX0	DCSET2	DCSET1	DCSET0	—
PGACS	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
ADRL	D7	D6	D5	D4	D3	D2	D1	D0
ADRM	D15	D14	D13	D12	D11	D10	D9	D8
ADRH	D23	D22	D21	D20	D19	D18	D17	D16
ADCR0	ADRST	ADSLP	ADOFF	ADOR2	ADOR1	ADOR0	—	VREFS
ADCR1	FLMS2	FLMS1	FLMS0	—	—	ADCDL	EOC	—
ADCS	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
DSOPC	DSOPEN	—	—	—	OPN1	OPN0	OPP1	OPP0
DSVCMC	—	—	—	—	OPSW3	OPSW2	OPSW1	OPSW0

A/D Converter Register List

Programmable Gain Amplifier Registers – PGAC0, PGAC1, PGACS

There are three registers related to the programmable gain control, PGAC0, PGAC1 and PGACS. The PGAC0 register is used to select the PGA gain, A/D converter gain and the A/D converter reference gain. As well, the PGAC1 register is used to define the input connection, differential input offset voltage adjustment control. In addition, The PGACS register is used to select the input ends for the PGA. Therefore, the input channels have to be determined by the CHSP[3:0] and CHSN[3:0] bits to determine which analog channel input pins, temperature sensor inputs or internal power supply are actually connected to the internal differential A/D converter.

• PGAC0 Register

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

Bit 7 Unimplemented, read as “0”

Bit 6~5 **VGS1~VGS0**: REFP/REFN Differential Reference Voltage Gain Selection
 00: VREFGN=1
 01: VREFGN=1/2
 10: VREFGN=1/4
 11: Reserved

Bit 4~3 **AGS1~AGS0**: A/D Converter PGAOP/PGAON Differential Input Signal Gain Selection
 00: ADGN=1
 01: ADGN=2
 10: ADGN=4
 11: Reserved

Bit 2~0 **PGS2~PGS0**: PGA DI+/DI- Differential Channel Input Gain Selection
 000: PGAGN=1
 001: PGAGN=2
 010: PGAGN=4
 011: PGAGN=8
 100: PGAGN=16
 101: PGAGN=32
 110: PGAGN=64
 111: PGAGN=128

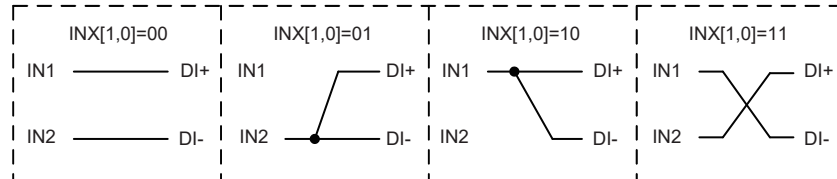
• **PGAC1 Register**

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

Bit 7 Unimplemented, read as “0”

Bit 6 **INIS**: The Selected Input Ends IN1 and IN2 Internal Connection Control Bit
 0: Not connected
 1: Connected

Bit 5~4 **INX1~INX0**: The Selected Input Ends, IN1/IN2 and the PGA Differential Input Ends, DI+/DI- Connection Control Bits



Bit 3~1 **DCSET2~DCSET0**: Differential Input Signal PGAOP/PGAON Offset Selection

000: DCSET=+0V
 001: DCSET=+0.25×ΔVR_I
 010: DCSET=+0.5×ΔVR_I
 011: DCSET=+0.75×ΔVR_I
 100: DCSET=+0V
 101: DCSET=-0.25×ΔVR_I
 110: DCSET=-0.5×ΔVR_I
 111: DCSET=-0.75×ΔVR_I

The voltage, ΔVR_I, is the differential reference voltage which is amplified by specific gain selection based on the selected inputs.

Bit 0 Unimplemented, read as “0”

• **PGACS Register**

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

Bit 7~4 **CHSN3~CHSN0**: Negative Input End IN2 Selection

0000: AN0
 0001: AN1
 0010: AN3
 0011: AN5
 0100: AN7
 0101: V_{IN}/6
 0110: LNOPO
 0111: V_{CM}
 1000: Temperature sensor output – V_{TSON}
 1001~1111: Reserved

These bits are used to select the negative input, IN2. If the IN2 input is selected as a single end input, the V_{CM} voltage must be selected as the positive input on IN1 for single end input applications. It is recommended that when the V_{TSON} signal is selected as the negative input, the V_{TSON} signal should be selected as the positive input for proper operations.

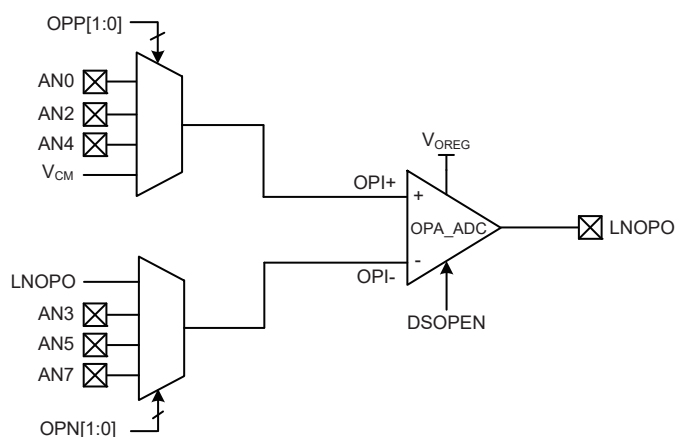
Bit 3~0 **CHSP3~CHSP0**: Positive Input End IN1 Selection

0000: AN0
 0001: AN2
 0010: AN4
 0011: AN6
 0100: $V_{IN}/5$
 0101: LNOPO
 0110: V_{CM}
 0111: Temperature sensor output – V_{TSOP}
 1000~1111: Reserved

These bits are used to select the positive input, IN1. If the IN1 input is selected as a single end input, the V_{CM} voltage must be selected as the negative input on IN2 for single end input applications. It is recommended that when the V_{TSOP} signal is selected as the positive input, the V_{TSOP} signal should be selected as the negative input for proper operations.

Low Noise Operational Amplifier Control Register – DSOPC

There is a fully integrated Operational Amplifier in the device. This OPA can be used for signal amplification according to specific user requirements. The OPA can be disabled or enabled entirely under software control using internal register. With specific control register, some OPA related applications can be more flexible and easier to be implemented, such as Unit Gain Buffer, Non-Inverting Amplifier, Inverting Amplifier and various kinds of filters, etc.



• DSOPC Register

Bit	7	6	5	4	3	2	1	0
Name	DSOPEN	—	—	—	OPN1	OPN0	OPP1	OPP0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **DSOPEN**: Operational amplifier control bit

0: Disable
 1: Enable

Bit 6~4 Unimplemented, read as “0”

Bit 3~2 **OPN1~OPN0**: Negative input end OPI- selection

00: LNOPO
 01: AN3
 10: AN5
 11: AN7

Bit 1~0 **OPP1~OPP0**: Positive input end OPI+ selection
 00: AN0
 01: AN2
 10: AN4
 11: V_{CM}

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

This device contains an internal 24-bit Delta Sigma A/D Converter, it requires three data registers to store the converted value. These are a high byte register, known as ADRH, a middle byte register, known as ADRM, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. D0~D23 are the A/D conversion result data bits.

• ADRL Register

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

“x”: Unknown

Bit 7~0 A/D Conversion Data Register bit 7~bit 0

• ADRM Register

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

“x”: Unknown

Bit 7~0 A/D Conversion Data Register bit 15~bit 8

• ADRH Register

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

“x”: Unknown

Bit 7~0 A/D Conversion Data Register bit 23~bit 16

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

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

• **ADCR0 Register**

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

- Bit 7 **ADRST**: A/D Converter Software Reset Control bit
0: Disable
1: Enable
This bit is used to reset the A/D converter internal digital SINC filter. This bit is set low for A/D normal operations. However, if set high, the internal digital SINC filter will be reset and the current A/D converted data will be aborted. A new A/D data conversion process will not be initiated until this bit is cleared to zero again.
- Bit 6 **ADSLP**: A/D Converter Sleep Mode Control bit
0: Normal mode
1: Sleep mode
This bit is used to determine whether the A/D converter enters the sleep mode or not when the A/D converter is powered on by clearing the ADOFF bit low. When the A/D converter is powered on and the ADSLP bit is low, the A/D converter will operate normally. However, the A/D converter will enter the sleep mode if the ADSLP bit is set high as the A/D converter has been powered on. The whole A/D converter circuit will be switched off except the PGA and internal Bandgap circuit to reduce the power consumption and V_{CM} start-up stable time.
- Bit 5 **ADOFF**: A/D Converter Module Power On/Off Control bit
0: Power on
1: Power off
This bit controls the power of the A/D converter module. This bit should be cleared to zero to enable the A/D converter. If the bit is set high then the A/D converter will be switched off reducing the device power consumption. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.
It is recommended to set the ADOFF bit high before the device enters the IDLE/ SLEEP mode for saving power. Setting the ADOFF bit high will power down the A/D converter module regardless of the ADSLP and ADRST bit settings.
- Bit 4 ~ 2 **ADOR2~ADOR0**: A/D Conversion Oversampling Rate Selection
000: Oversampling rate $OSR=16384$
001: Oversampling rate $OSR=8192$
010: Oversampling rate $OSR=4096$
011: Oversampling rate $OSR=2048$
100: Oversampling rate $OSR=1024$
101: Oversampling rate $OSR=512$
110: Oversampling rate $OSR=256$
111: Oversampling rate $OSR=128$
- Bit 1 Unimplemented, read as “0”
- Bit 0 **VREFS**: A/D Converter Reference Voltage Pair Selection
0: Internal reference voltage pair – V_{CM} & AV_{SS}
1: Internal reference voltage pair – V_{OREG} & AV_{SS}

• **ADCR1 Register**

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

- Bit 7~5 **FLMS2~FLMS0**: A/D Converter Clock f_{ADCK} Selection and Sampled Data Doubling Function (CHOP) Enable Control
 000: CHOP=2, $f_{ADCK}=f_{MCLK}/30$
 010: CHOP=2, $f_{ADCK}=f_{MCLK}/12$
 100: CHOP=1, $f_{ADCK}=f_{MCLK}/30$
 110: CHOP=1, $f_{ADCK}=f_{MCLK}/12$
 Others: Reserved
 When the CHOP bit is equal to 2, it means that the sampled data amount will be doubled for the normal conversion mode. However, it can be regarded as the low latency conversion mode if the CHOP bit is equal to 1, which means that the sampled data doubling function is disabled.
- Bit 4~3 Unimplemented, read as “0”
- Bit 2 **ADCDL**: A/D Converted Data Latch Function Enable Control
 0: Disable data latch function
 1: Enable data latch function
 If the A/D converted data latch function is enabled, the latest converted data value will be latched and not be updated by any subsequent converted results until this function is disabled. Although the converted data is latched into the data registers, the A/D converter circuits remain operational, but will not generate interrupt and EOC will not change. It is recommended that this bit should be set high before reading the converted data in the ADRL, ADRM and ADRH registers. After the converted data has been read out, the bit can then be cleared to low to disable the A/D converter data latch function and allow further conversion values to be stored. In this way, the possibility of obtaining undesired data during A/D converter conversions can be prevented.
- Bit 1 **EOC**: End of A/D Conversion Flag
 0: A/D conversion in progress
 1: A/D conversion ended
 This bit must be cleared by software.
- Bit 0 Unimplemented, read as “0”

• **ADCS Register**

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

- Bit 7~5 Unimplemented, read as “0”
- Bit 4~0 **ADCK4~ADCK0**: A/D Converter Clock Source f_{MCLK} setup
 00000~11110: $f_{MCLK}=f_{SYS}/2 / (ADCK[4:0]+1)$
 11111: $f_{MCLK}=f_{SYS}$

A/D Operation

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

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

Note: 1. The V_{CM} generator can be switched on or off by the bandgap on or off.

2. The Temperature sensor can be switched on or off by configuring the CHSN[3:0] or CHSP[3:0] bits.

3. “x” means unknown.

A/D Operation Mode Selection

To enable the A/D converter, the first step is to disable the A/D converter power down and sleep mode by clearing the ADOFF and ADSLP bits to make sure the A/D converter is powered up. The ADRST bit in the ADCR0 register is used to start and reset the A/D converter after power on. To set ADRST bit from low to high and then low again, a conversion cycle of the 1-bit analog-to-digital converted data will be initiated in the SINC filter. After this setup is complete, the A/D Converter is ready for operation. These three bits are used to control the overall start operation of the internal analog to digital converter.

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

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

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

Summary of A/D Conversion Steps

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

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

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

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by setting bit ADOFF high in the ADCR0 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.

A/D Transfer Function

This device contains a 24-bit Delta Sigma A/D converter and its full-scale converted digitised value is from 8388607 to -8388608 in decimal value. The converted data format is formed by a two's complement binary value. The MSB of the converted data is the signed bit. Since the full-scale analog input value is equal to the amplified value of the V_{CM} or differential reference input voltage, ΔVR_I , selected by the VREFS bit in ADCR0 register, this gives a single bit analog input value of ΔVR_I divided by 8388608.

$$1 \text{ LSB} = \Delta VR_I / 8388608$$

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

$$\Delta SI_I = (PGAGN \times ADGN \times \Delta DI_{\pm}) + DCSET$$

$$\Delta VR_I = VREFGN \times \Delta VR_{\pm}$$

$$ADC_Conversion_Data = (\Delta SI_I / \Delta VR_I) \times K$$

Where K is equal to 2^{23}

Note:

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

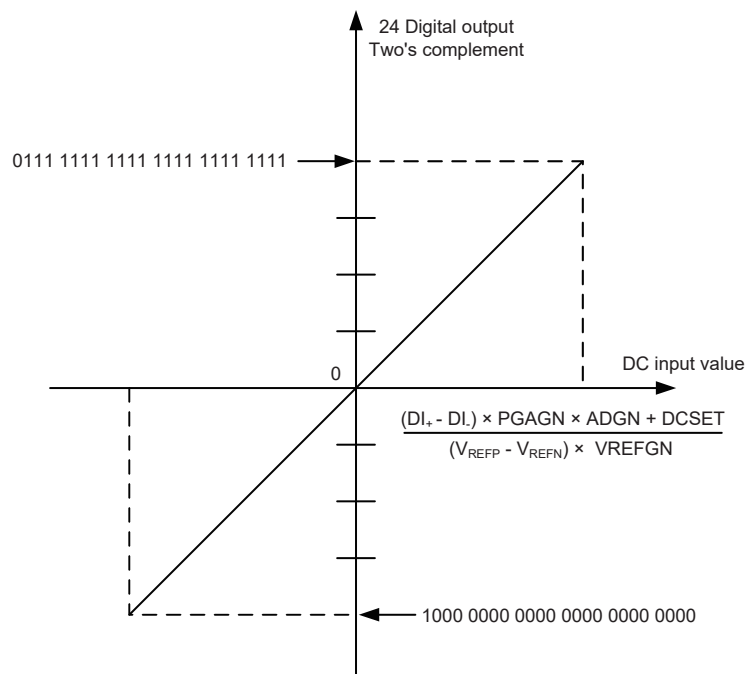
Due to the digital system design of the Delta Sigma A/D Converter, the maximum number of the A/D converted value is 8388607 and the minimum value is -8388608, therefore, we can have the middle number 0. The ADC_Conversion_Data equation illustrates this range of converted data variation.

A/D conversion data (2's compliment, Hexadecimal)	Decimal Value
0x7FFFFFFF	8388607
0x800000	-8388608

A/D Conversion Data Range

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

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



A/D Converted Data

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

A/D Converted Data to Voltage

The designer can recover the converted data by the following equations:

If MSB=0 (Positive Converted data):

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

If MSB=1 (Negative Converted data):

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

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

A/D Programming Example

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

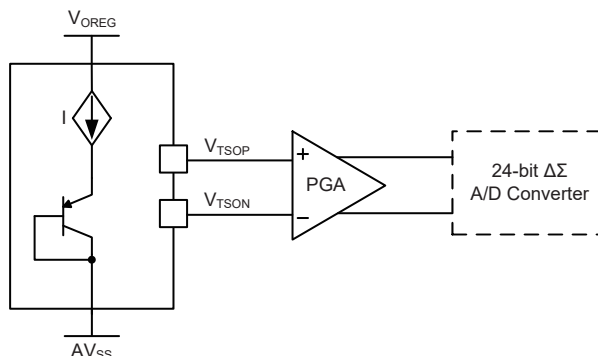
```
#include bh67f2752.inc
data .section 'data'
    adc_result_data_l db ?
    adc_result_data_m db ?
    adc_result_data_h db ?
code .section 'code'
start:
    clr ADE                ; Disable A/D converter interrupt
    mov a, 083H            ; Power control for PGA, A/D converter
    mov PWRC, a            ; PWRC=10000011, LDO enable, LDO Bypass disable,
                          ; LDO output voltage: 3.3V

    mov a, 000H
    mov PGAC0, a           ; PGA gain=1, A/D converter gain=1, VREF gain=1
    mov a, 000H
    mov PGAC1, a           ; INIS, INX, DCSET in default value
    set VREFS              ; for using internal reference voltage pair VOREG & AVSS
    clr ADOR2              ; for 10Hz output data rate, ADOR[2:0]=001, FLMS[2:0]=000
    clr ADOR1
    set ADOR0
    clr FLMS2
    clr FLMS1
    clr FLMS0
    clr ADOFF              ; A/D converter exit power down mode.
    set ADRST              ; A/D converter in reset mode
    clr ADRST              ; A/D converter in conversions (continuous mode)
    clr EOC                ; Clear "EOC" flag

loop:
    snz EOC                ; Polling "EOC" flag
    jmp loop               ; Wait for read data
    clr adc_result_data_h
    clr adc_result_data_m
    clr adc_result_data_l
    mov a, ADRL
    mov adc_result_data_l, a ; Get Low byte A/D converter value
    mov a, ADRM
    mov adc_result_data_m, a ; Get Middle byte A/D converter value
    mov a, ADRH
    mov adc_result_data_h, a ; Get High byte A/D converter value
get_adc_value_ok:
    clr EOC                ; Clearing read flag
    jmp loop               ; for next data read
end
```


Temperature sensor

This device provides an internal temperature sensor to compensate the device due to temperature effects. By selecting the PGA input channels as V_{TSOP} and V_{TSON} , the A/D Converter can obtain temperature information allowing compensation to be made to the A/D converted data.



UART Interface

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

The integrated UART function contains the following features:

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

UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as TX and RX. The TX and RX pins are the UART transmitter and receiver pins respectively. Along

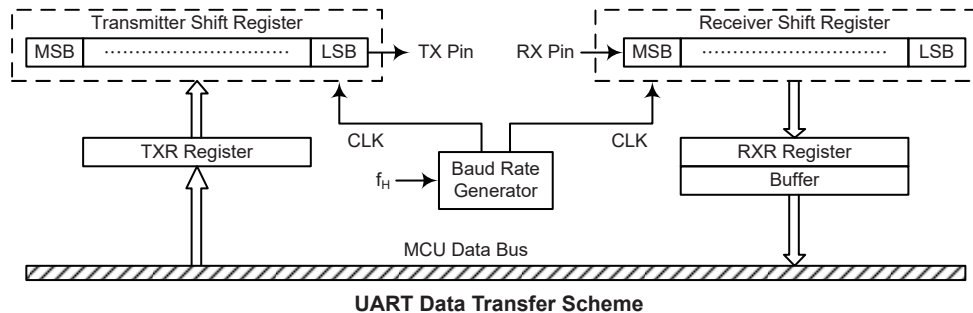
with the UARTEN bit, the TXEN and RXEN bits, if set, will automatically setup these I/O or other pin-shared functional pins to their respective TX output and RX input conditions and disable any pull-high resistor option which may exist on the TX or RX pins. When the TX or RX pin function is disabled by clearing the UARTEN and TXEN or RXEN bit, the TX or RX pin can be used as a general purpose I/O or other pin-shared functional pin.

UART Data Transfer Scheme

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

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

It should be noted that the actual register for data transmission and reception, although referred to in the text, and in application programs, as separate TXR and RXR registers, only exists as a single shared register in the Data Memory. This shared register known as the TXR_RXR register is used for both data transmission and data reception.



UART Status and Control Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
TXR_RXR	D7	D6	D5	D4	D3	D2	D1	D0
BRG	D7	D6	D5	D4	D3	D2	D1	D0

UART Register List

• USR Register

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

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

Bit 7 **PERR:** Parity Error Flag

- 0: No parity error is detected
- 1: Parity error is detected

The PERR flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if Parity mode (odd or even) is selected. The flag can also be cleared by a software sequence which involves a read to the status register USR followed by an access to the RXR data register.

Bit 6 **NF:** Noise Flag

- 0: No noise is detected
- 1: Noise is detected

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

Bit 5 **FERR:** Framing Error Flag

- 0: No framing error is detected
- 1: Framing error is detected

The FERR flag is the framing error flag. When this read only flag is “0”, it indicates that there is no framing error. When the flag is “1”, it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register USR followed by an access to the RXR data register.

Bit 4 **OERR:** Overrun Error Flag

- 0: No overrun error is detected
- 1: Overrun error is detected

The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is “0”, it indicates that there is no overrun error. When the flag is “1”, it indicates that an overrun error occurs which will inhibit further transfers to the RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register USR followed by an access to the RXR data register.

Bit 3 **RIDLE:** Receiver Status

- 0: Data reception is in progress (data being received)
- 1: No data reception is in progress (receiver is idle)

The RIDLE flag is the receiver status flag. When this read only flag is “0”, it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is “1”, it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is “1” indicating that the UART receiver is idle and the RX pin stays in logic high condition.

- Bit 2 RXIF: Receive RXR Data Register Status**
 0: RXR data register is empty
 1: RXR data register has available data
 The RXIF flag is the receive data register status flag. When this read only flag is “0”, it indicates that the RXR read data register is empty. When the flag is “1”, it indicates that the RXR read data register contains new data. When the contents of the shift register are transferred to the RXR register, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag is cleared when the USR register is read with RXIF set, followed by a read from the RXR register, and if the RXR register has no data available.
- Bit 1 TIDLE: Transmission Idle**
 0: Data transmission is in progress (data being transmitted)
 1: No data transmission is in progress (transmitter is idle)
 The TIDLE flag is known as the transmission complete flag. When this read only flag is “0”, it indicates that a transmission is in progress. This flag will be set to “1” when the TXIF flag is “1” and when there is no transmit data or break character being transmitted. When TIDLE is equal to “1”, the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared by reading the USR register with TIDLE set and then writing to the TXR register. The flag is not generated when a data character or a break is queued and ready to be sent.
- Bit 0 TXIF: Transmit TXR Data Register Status**
 0: Character is not transferred to the transmit shift register
 1: Character has transferred to the transmit shift register (TXR data register is empty)
 The TXIF flag is the transmit data register empty flag. When this read only flag is “0”, it indicates that the character is not transferred to the transmitter shift register. When the flag is “1”, it indicates that the transmitter shift register has received a character from the TXR data register. The TXIF flag is cleared by reading the UART status register (USR) with TXIF set and then writing to the TXR data register. Note that when the TXEN bit is set, the TXIF flag bit will also be set since the transmit data register is not yet full.

• UCR1 Register

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

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

“x”: Unknown

- Bit 7 UARTEN: UART Function Enable Control**
 0: Disable UART. TX and RX pins are in a floating state
 1: Enable UART. TX and RX pins function as UART pins
 The UARTEN bit is the UART enable bit. When this bit is equal to “0”, the UART will be disabled and the RX pin as well as the TX pin will be set in a floating state. When the bit is equal to “1”, the UART will be enabled and the TX and RX pins will function as defined by the TXEN and RXEN enable control bits.
 When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2 and BRG registers will remain unaffected. If the UART is active and the UARTEN bit is

cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

- Bit 6 **BNO**: Number of Data Transfer Bits Selection
 0: 8-bit data transfer
 1: 9-bit data transfer
 This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to “1”, a 9-bit data length format will be selected. If the bit is equal to “0”, then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits RX8 and TX8 will be used to store the 9th bit of the received and transmitted data respectively.
- Bit 5 **PREN**: Parity Function Enable Control
 0: Parity function is disabled
 1: Parity function is enabled
 This is the parity enable bit. When this bit is equal to “1”, the parity function will be enabled. If the bit is equal to “0”, then the parity function will be disabled.
- Bit 4 **PRT**: Parity Type Selection Bit
 0: Even parity for parity generator
 1: Odd parity for parity generator
 This bit is the parity type selection bit. When this bit is equal to “1”, odd parity type will be selected. If the bit is equal to “0”, then even parity type will be selected.
- Bit 3 **STOPS**: Number of Stop Bits Selection
 0: One stop bit format is used
 1: Two stop bits format is used
 This bit determines if one or two stop bits are to be used for the TX pin. When this bit is equal to “1”, two stop bits are used. If this bit is equal to “0”, then only one stop bit is used.
- Bit 2 **TXBRK**: Transmit Break Character
 0: No break character is transmitted
 1: Break characters transmit
 The TXBRK bit is the Transmit Break Character bit. When this bit is “0”, there are no break characters and the TX pin operates normally. When the bit is “1”, there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to “1”, after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.
- Bit 1 **RX8**: Receive Data Bit 8 for 9-bit Data Transfer Format (read only)
 This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.
- Bit 0 **TX8**: Transmit Data Bit 8 for 9-bit Data Transfer Format (write only)
 This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• UCR2 Register

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

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

Bit 7 **TXEN**: UART Transmitter Enable Control

- 0: UART transmitter is disabled
- 1: UART transmitter is enabled

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

Bit 6 **RXEN**: UART Receiver Enable Control

- 0: UART receiver is disabled
- 1: UART receiver is enabled

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

Bit 5 **BRGH**: Baud Rate Speed Selection

- 0: Low speed baud rate
- 1: High speed baud rate

The bit named BRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register BRG, controls the Baud Rate of the UART. If this bit is equal to “1”, the high speed mode is selected. If the bit is equal to “0”, the low speed mode is selected.

Bit 4 **ADDEN**: Address Detect Function Enable Control

- 0: Address detection function is disabled
- 1: Address detection function is enabled

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

- Bit 3 WAKE:** RX Pin Falling Edge Wake-up UART Function Enable Control
0: RX pin wake-up UART function is disabled
1: RX pin wake-up UART function is enabled
This bit is used to control the wake-up UART function when a falling edge on the RX pin occurs. Note that this bit is only available when the UART clock source (f_H) is switched off. There will be no RX pin wake-up UART function if the UART clock (f_H) exists. If the WAKE bit is set to 1 as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the RX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function can not resume even if there is a falling edge on the RX pin when the WAKE bit is cleared to 0.
- Bit 2 RIE:** Receiver Interrupt Enable Control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled
This bit enables or disables the receiver interrupt. If this bit is equal to “1” and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.
- Bit 1 TIE:** Transmitter Idle Interrupt Enable Control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled
This bit enables or disables the transmitter idle interrupt. If this bit is equal to “1” and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.
- Bit 0 TEIE:** Transmitter Empty Interrupt Enable Control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled
This bit enables or disables the transmitter empty interrupt. If this bit is equal to “1” and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

• TXR_RXR Register

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

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

“x”: Unknown

Bit 7~0 D7~D0: UART Transmit/Receive Data bit 7 ~ bit 0

Baud Rate Generator

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

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

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

• BRG Register

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

"x": Unknown

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

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

Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, and with BRGH set to "0" determine the BRG register value N, the actual baud rate and the error value for a desired baud rate of 4800.

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

Re-arranging this equation gives $N = [f_H / (BR \times 64)] - 1$

Giving a value for $N = [4000000 / (4800 \times 64)] - 1 = 12.0208$

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

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

UART Setup and Control

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

Enabling/Disabling the UART Interface

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

Clearing the UARTEN bit to zero will disable the TX and RX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins. When the UART function is disabled the

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

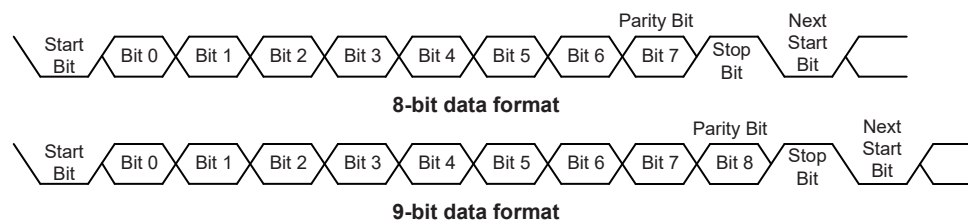
Data, Parity and Stop Bit Selection

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

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

Transmitter Receiver Data Format

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



UART Transmitter

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

should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN bit is set, but the data will not be transmitted until the TXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin will then return to the I/O or other pin-shared function.

Transmitting Data

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

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

This sequence of events can now be repeated to send additional data. It should be noted that when TXIF is “0”, data will be inhibited from being written to the TXR register. Clearing the TXIF flag is always achieved using the following software sequence:

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

The read-only TXIF flag is set by the UART hardware and if set indicates that the TXR register is empty and that other data can now be written into the TXR register without overwriting the previous data. If the TEIE bit is set then the TXIF flag will generate an interrupt. During a data transmission, a write instruction to the TXR register will place the data into the TXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLE bit the following software sequence is used:

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

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

Transmit Break

If the TXBRK bit is set then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by $13 \times N$ ‘0’ bits and stop bits, where $N=1, 2$, etc. If a break character is to be transmitted then the TXBRK bit must be first set by the application program and then cleared to generate the stop bits. Transmitting a break character will not generate a transmit

interrupt. Note that a break condition length is at least 13 bits long. If the TXBRK bit is continually kept at a logic high level then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the TXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out one or two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

UART Receiver

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

Receiving Data

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

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

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

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

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

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

1. A USR register access
2. An RXR register read execution

Receive Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the

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

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

Idle Status

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

Receiver Interrupt

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

Managing Receiver Errors

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

Overrun Error – OERR Flag

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

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

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

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

Noise Error – NF Flag

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

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

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

Framing Error – FERR Flag

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

Parity Error – PERR Flag

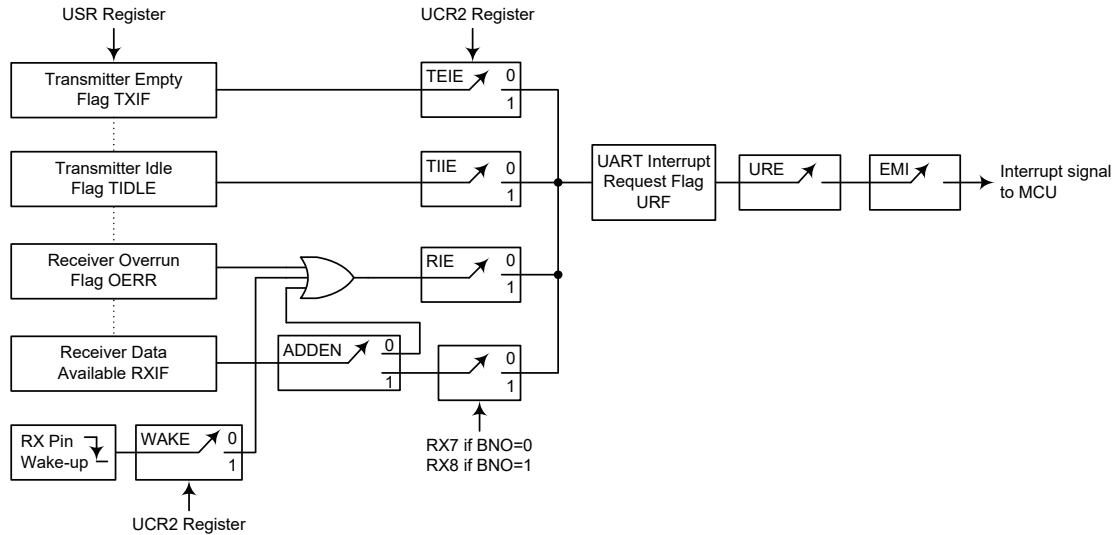
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN bit is “1”, and if the parity type, odd or even is selected. The read only PERR flag and the received data will be recorded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

UART Module Interrupt Structure

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

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the UART clock source (f_{H1}) is switched off and the WAKE and RIE bits in the UCR2 register are set. Note that in the event of an RX wake-up interrupt occurring, there will be a certain period of delay, commonly known as the System Start-up Time, for the oscillator to restart and stabilize before the system resumes normal operation.

Note that the USR register flags are read only and cannot be cleared or set by the application program, neither will they be cleared when the program jumps to the corresponding interrupt servicing routine, as is the case for some of the other interrupts. The flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART register section. The overall UART interrupt can be disabled or enabled by the related interrupt enable control bits in the interrupt control registers of the microcontroller to decide whether the interrupt requested by the UART module is masked out or allowed.



UART Interrupt Scheme

Address Detect Mode

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is “1”, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the related interrupt enable control bit URE and the EMI bit must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO bit is “1” or the 8th bit if BNO bit is “0”. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is “0”, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to zero.

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

ADDEN Bit Function

UART Power Down and Wake-up

When the UART clock (f_H) is switched off, the UART will cease to function. If the UART clock (f_H) is switched off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the device enters the IDLE or SLEEP mode while receiving data, then the reception of data will likewise be paused. When the device enters the IDLE or SLEEP mode, note that the USR, UCR1, UCR2, transmit and receive registers, as well as the BRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the UART clock (f_{H1}) is off, then a falling edge on the RX pin will trigger an RX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX pin will be ignored.

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

Serial Peripheral Interface – SPI

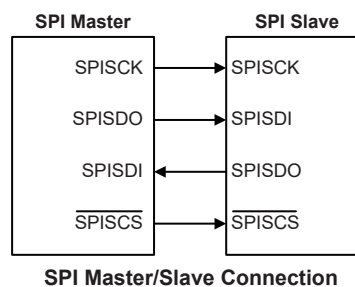
The device contains an independent SPI function. The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively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

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

SPI Interface Operation

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

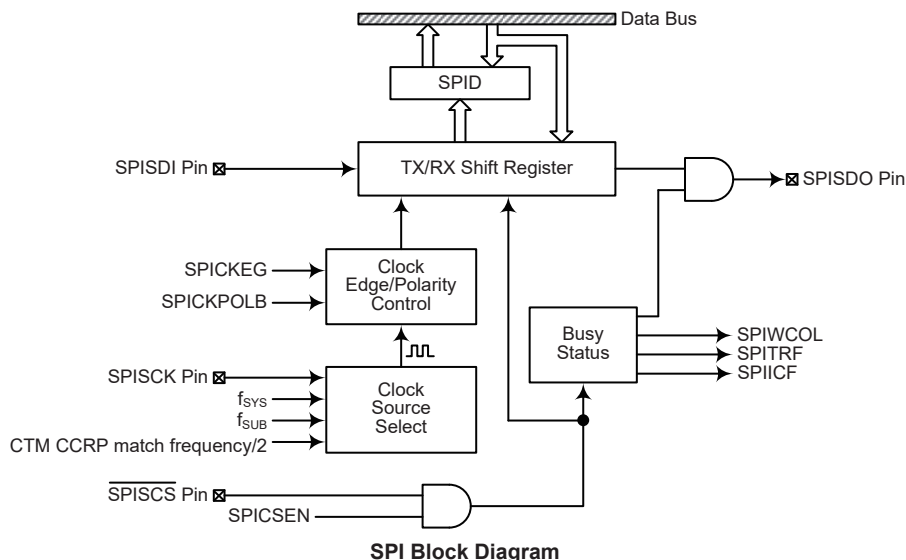
The $\overline{\text{SPISCS}}$ pin is controlled by the application program, set the SPICSEN bit to “1” to enable the $\overline{\text{SPISCS}}$ pin function and clear the SPICSEN bit to “0” to place the $\overline{\text{SPISCS}}$ pin into a floating state.



The SPI function in the device offers the following features:

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

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



SPI Registers

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

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

SPI Register List

SPI Data Register

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

• **SPID Register**

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

“x”: Unknown

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

SPI Control Registers

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

• **SPIC0 Register**

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

Bit 7~5 **SPIM2~SPIM0**: SPI Operating Mode Control

000: SPI master mode; SPI clock is $f_{SYS}/4$
 001: SPI master mode; SPI clock is $f_{SYS}/16$
 010: SPI master mode; SPI clock is $f_{SYS}/64$
 011: SPI master mode; SPI clock is f_{SUB}
 100: SPI master mode; SPI clock is CTM1 CCRP match frequency/2
 101: SPI slave mode
 110: SPI disabled
 111: SPI disabled

These bits are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from CTM1 and f_{SUB} . If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4~2 Unimplemented, read as “0”

Bit 1 **SPIEN**: SPI Enable Control

0: Disable
 1: Enable

The bit is the overall on/off control for the SPI interface. When the SPIEN bit is cleared to zero to disable the SPI interface, the SPISDI, SPISDO, SPISCK and SPISCS lines will lose their SPI function and the SPI operating current will be reduced to a minimum value. When the bit is high the SPI interface is enabled.

Bit 0 **SPIICF**: SPI Incomplete Flag

0: SPI incomplete condition is not occurred
 1: SPI incomplete condition is occurred

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

• **SPIC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SPICKPOLB	SPICKEG	SPIMLS	SPICSEN	SPIWCOL	SPITRF
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5 **SPICKPOLB**: SPI clock line base condition selection

0: The SPISCK line will be high when the clock is inactive

1: The SPISCK line will be low when the clock is inactive

The SPICKPOLB bit determines the base condition of the clock line, if the bit is high, then the SPISCK line will be low when the clock is inactive. When the SPICKPOLB bit is low, then the SPISCK line will be high when the clock is inactive.

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

SPICKPOLB=0

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

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

SPICKPOLB=1

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

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

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

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

0: LSB first

1: MSB first

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

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

0: Disable

1: Enable

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

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

0: No collision

1: Collision

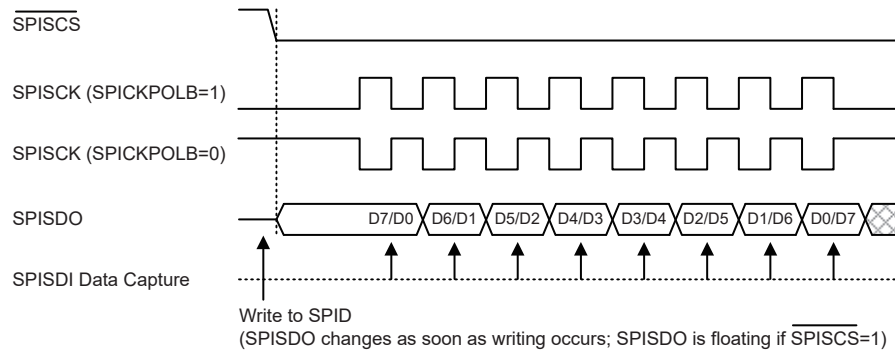
The SPIWCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SPID register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared to 0 by the application program.

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

0: SPI data is being transferred

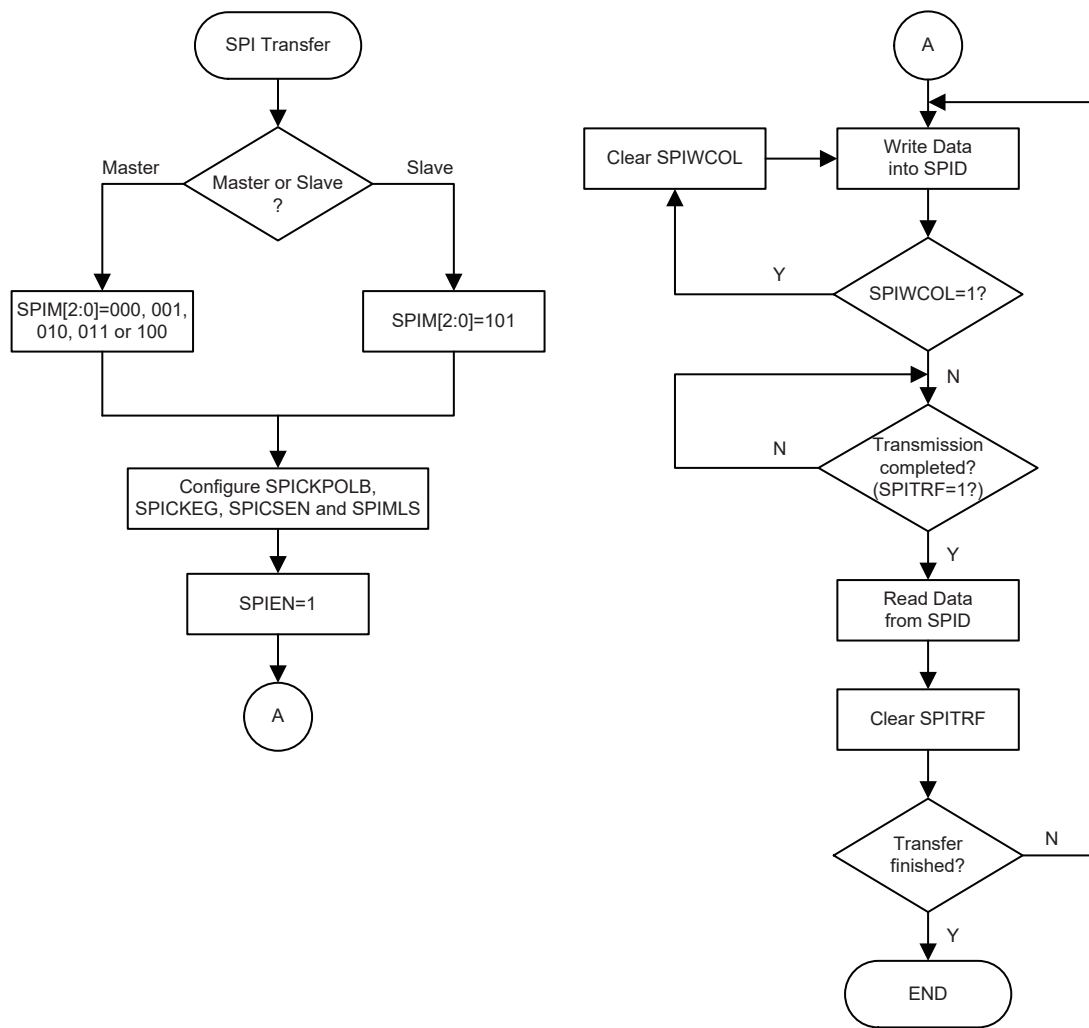
1: SPI data transmission is completed

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



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

SPI Slave Mode Timing – SPICKEG=1



SPI Transfer Control Flowchart

SPI Bus Enable/Disable

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

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

SPI Operation

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

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

Master Mode:

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

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

Slave Mode:

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

Error Detection

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

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 device contains 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.

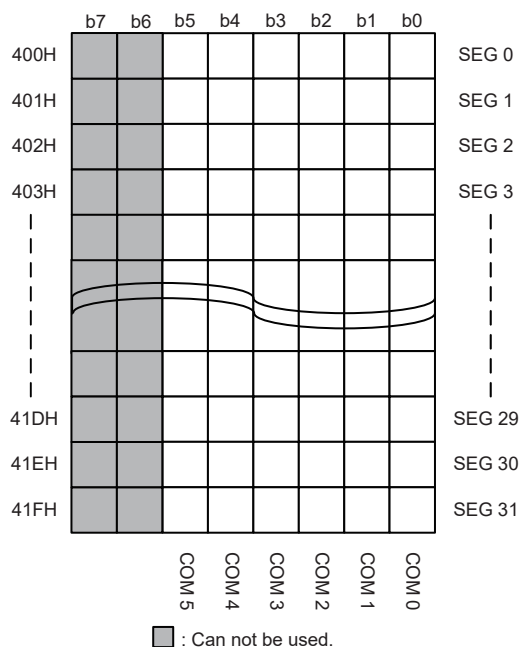
Driver No.	Duty	Bias	Bias Type	Waveform Type
32×4	1/4	1/3	R	A or B
30×6	1/6	1/3	R	A or B

LCD Data 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 is stored in its own independent Sector 4 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 4 is selected by writing a value of “04H” 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 4 selected, then using MP1L or MP2L to read or write to the memory area, starting with address “00H” for all the devices, will result in operations to the LCD Memory. Directly addressing the LCD Display Memory can be applicable using the extended instructions for the full range address access.

The accompanying LCD Memory Map diagrams shows how the internal LCD Memory is mapped to the Segments and Commons of the display for the device. It should be noted that the unused LCD RAM cannot be used as general purpose data memory for application. For example, if the LCD is selected as 1/4 duty, the COM4~COM5 will be read as “0” only.



LCD Memory Map

LCD Clock Source

The LCD clock source is the internal clock signal, f_{SUB} , divided by 8 using an internal divider circuit. The f_{SUB} internal clock is supplied by the LIRC oscillator. For proper LCD operation, this arrangement is provided to generate an ideal LCD clock source frequency of 4kHz.

LCD Register

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

Various bits in the LCDC0 register control functions such as LCD waveform type, bias current selection together with the overall LCD enable/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 FAST, SLOW or IDLE Mode. If the device is in the SLEEP Mode then the display will always be disabled. The TYPE bit in the LCDC0 register is used to select whether Type A or Type B LCD waveform signals are used.

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
LCDC0	TYPE	—	—	—	—	LCDIS1	LCDIS0	LCDEN
LCDCP	—	—	—	DTYC	LCDPR	—	CPVS1	CPVS0

LCD Register List

• **LCDC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	TYPE	—	—	—	—	LCDIS1	LCDIS0	LCDEN
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TYPE**: LCD waveform type selection

0: Type A

1: Type B

Bit 6~3 Unimplemented, read as “0”

Bit 2~1 **LCDIS1~LCDIS0**: LCD Bias Current Selection ($V_A = V_{PLCD} = V_{DD}$, 1/3 bias)

00: 25 μ A

01: 50 μ A

10: 100 μ A

11: 200 μ A

Bit 0 **LCDEN**: LCD Enable Control

0: Disable

1: Enable

In the FAST, SLOW or IDLE mode, the LCD on/off function can be controlled by this bit. However, in the SLEEP mode, the LCD function is always switched off.

• **LCDCP Register**

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

Bit 7~5 Unimplemented, read as “0”

Bit 4 **DTYC**: Define LCD duty

0: 1/4 Duty

1: 1/6 Duty

If LCD selects 1/4 duty, the COM0~COM3 pins will be used as LCD COM output pin, the COM4~COM5 pins can then be configured as other pin-shared functions. If 1/6 duty is selected, all the COM pins can be used for LCD COM output.

Bit 3 **LCDPR**: LCD Power selection

0: PLCD pin

1: Internal charge pump

When the LCDPR bit is cleared to “0”, the LCD power will be derived from the PLCD pin and internal charge pump circuit will be disabled.

Bit 2 Unimplemented, read as “0”

Bit 1~0 **CPVS1~CPVS0**: Charge pump output voltage selection

00: 3.3V

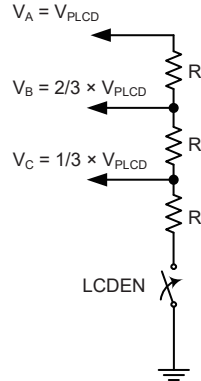
01: 3.0V

10: 2.7V

11: 4.5V

LCD Voltage Source and Biasing

For the 1/3 bias scheme, four voltage levels V_{SS} , V_A , V_B and V_C are utilised. The voltage V_A can be powered up by PLCD pin or internal charge pump regulator, selected by the LCDPR bit in the LCDCP register. There are four kinds of the internal charge pump voltage output, determined by the CPVS[1:0] bits in the LCDCP register. The voltage V_B is equal to $V_A \times 2/3$ while V_C is equal to $V_A \times 1/3$.

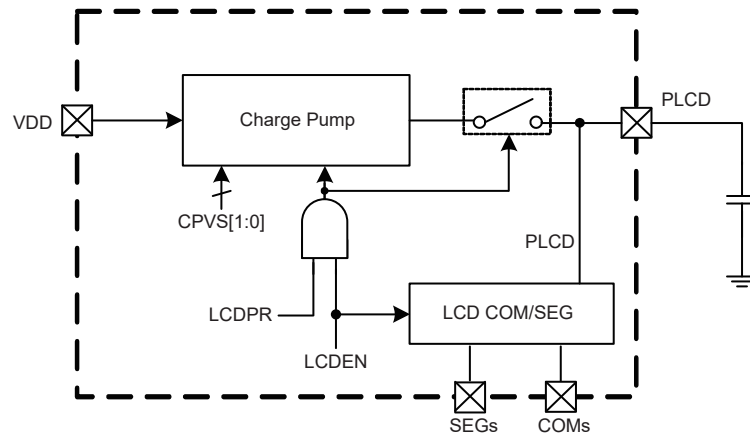


- Note: 1. When $LCDPR=1$, the $PLCD$ pin should externally connect a $4.7\mu F$ capacitor; when $LCDPR=0$, the $PLCD$ pin does not require an external capacitor.
 2. The DC path will be switched off when the LCD is disabled.
 3. The internal charge pump will only be activated when the $LCDEN$ and $LCDPR$ bits are both set high.

LCD Bias Configurations ($LCDPR=0$)

LCD Charge Pump

The COMs and SEGs pins can be powered up by the $PLCD$ pin input or the internal charge pump regulator, selected by the $LCDPR$ bit in the $LCDCP$ register. When the $LCDPR$ bit is set low, the LCD driver power is supplied by the external $PLCD$ pin. If the $LCDPR$ bit is set high, the LCD driver power is supplied by the internal charge pump circuit. There are four charge pump output voltage levels which are selected by the $CPVS1 \sim CPVS0$ bits in the $LCDCP$ register. If the internal charge pump circuit is used, an external $4.7\mu F$ capacitor should be connected to the external $PLCD$ pin for output voltage stability. The charge pump will only be activated when the $LCDEN$ and $LCDPR$ bits are both set high.



LCD Driver Charge Pump Circuit

LCDPR	CPVS[1:0]	LCD Power Supply
0	xx	From PLCD pin
1	00	Charge Pump Circuit output, 3.3V
	01	Charge Pump Circuit output, 3.0V
	10	Charge Pump Circuit output, 2.7V
	11	Charge Pump Circuit output, 4.5V

LCD Driver Power Supply

LCD Reset Status

The LCD has an internal reset function that is an OR function of the inverted LCDEN bit in the LCDC0 register and the SLEEP function. Clearing the LCDEN bit to zero will reset the LCD function. The LCD function will also be reset after the device enters the SLEEP mode even if the LCDEN bit is set to “1” to enable the LCD driver function.

When the LCDEN bit is set to “1” to enable the LCD driver and then an MCU reset occurs, the LCD driver will be reset and the COM and SEG output will be in a floating state during the MCU reset duration. The reset operation will take a time of $t_{RSTD} + t_{SST}$. Refer to the System Start Up Time Characteristics for t_{RSTD} and t_{SST} details.

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

Note:1. The watchdog time-out reset in the IDLE or SLEEP Mode is excluded from the MCU Reset conditions.

2. “x”: Don’t care.

LCD Reset Status

LCD Driver Output

The output structure of the LCD driver can be 32×4 or 30×6. The LCD driver bias type is R type and has a fixed bias value of 1/3.

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

The requirement to limit the DC voltage to zero and to control as many pixels as possible with a minimum number of connections requires that both a time and amplitude signal is generated and applied to the application LCD. These time and amplitude varying signals are automatically generated by the LCD driver circuits in the microcontroller. What is known as the duty determines the number of common lines used, which are also known as backplanes or COMs. For example, the duty, which is to have 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.

4 COMs, 1/3 Bias

LCD Display Off Mode

COM0 ~ COM3

All segment outputs

Normal Operation Mode

1 Frame

COM0

COM1

COM2

COM3

All segments are OFF

COM0 side segments are ON

COM1 side segments are ON

COM2 side segments are ON

COM3 side segments are ON

COM0,1 side segments are ON

COM0,2 side segments are ON

COM0,3 side segments are ON

(other combinations are omitted)

All segments are ON

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

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



6 COMs, 1/3 Bias

LCD Display Off Mode

COM0 ~ COM5

All sement outputs

Normal Operation Mode

COM0

COM1

COM2

COM3

COM4

COM5

All segments are OFF

COM0 side segments are ON

COM1 side segments are ON

COM2 side segments are ON

COM3 side segments are ON

COM4 side segments are ON

COM5 side segments are ON

COM0,1 side segments are ON

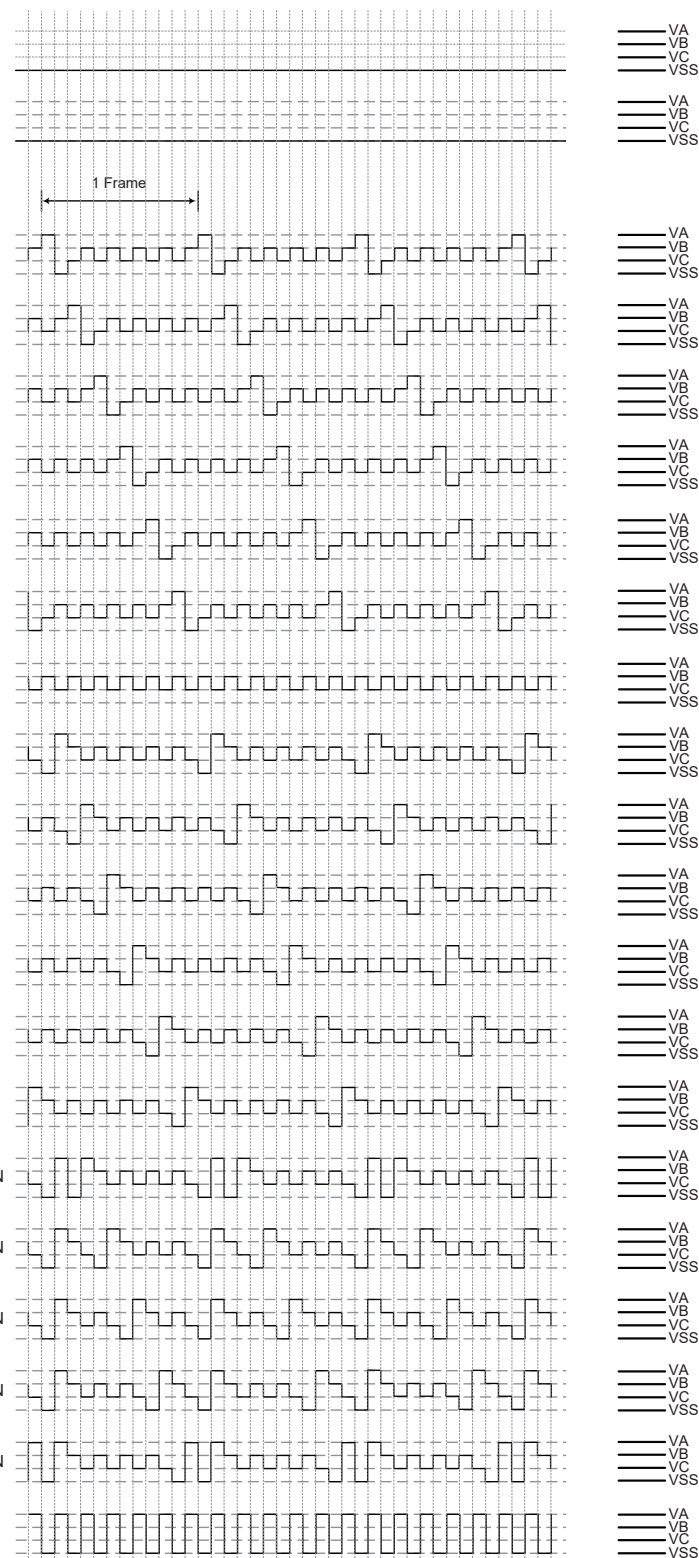
COM0,2 side segments are ON

COM0,3 side segments are ON

COM0,4 side segments are ON

COM0,5 side segments are ON

All sement outputs are ON



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

LCD Display Off Mode

COM0 ~ COM5

All segment outputs

Normal Operation Mode

1 Frame

COM0

COM1

COM2

COM3

COM4

COM5

All segments are OFF

COM0 side segments are ON

COM1 side segments are ON

COM2 side segments are ON

COM3 side segments are ON

COM4 side segments are ON

COM5 side segments are ON

COM0,1 side segments are ON

COM0,2 side segments are ON

COM0,3 side segments are ON

COM0,4 side segments are ON

COM0,5 side segments are ON

All segments are ON

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

VA
VB
VC
VSS

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

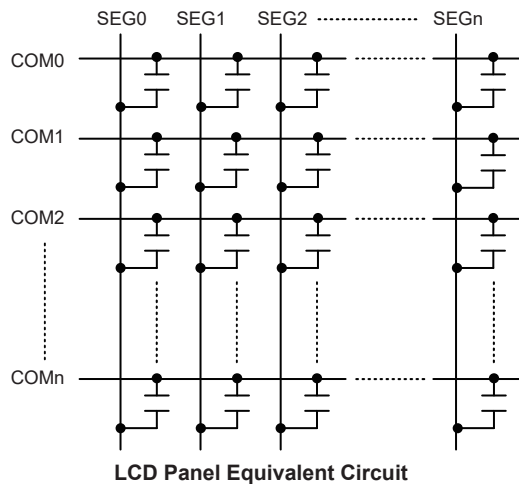
Programming Considerations

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.



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 interrupt functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the Timer Modules (TM), Time Bases, Low Voltage Detector (LVD), EEPROM, UART, SPI and the A/D converter.

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. The registers fall into three categories. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0~MFI2 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupts trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pin	INTnE	INTnF	n=0~1
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~2
Time Base	TBnE	TBnF	n=0~1
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
UART	URE	URF	—
SPI	SPIE	SPIF	—
CTM	CTMnPE	CTMnPF	n=0~1
	CTMnAE	CTMnAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
INTC1	URF	MF2F	MF1F	MF0F	URE	MF2E	MF1E	MF0E
INTC2	—	TB1F	TB0F	SPIF	—	TB1E	TB0E	SPIE
MFI0	—	—	CTM0AF	CTM0PF	—	—	CTM0AE	CTM0PE
MFI1	—	—	CTM1AF	CTM1PF	—	—	CTM1AE	CTM1PE
MFI2	—	—	DEF	LVF	—	—	DEE	LVE

Interrupt Register List

• **INTEG Register**

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

- Bit 7~4 Unimplemented, read as “0”
- Bit 3~2 **INT1S1~INT1S0**: Interrupt Edge Control for INT1 Pin
 00: Disable
 01: Rising edge
 10: Falling edge
 11: Rising and falling edges
- Bit 1~0 **INT0S1~INT0S0**: Interrupt Edge Control for INT0 Pin
 00: Disable
 01: Rising edge
 10: Falling edge
 11: Rising and falling edges

• **INTC0 Register**

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

- Bit 7 Unimplemented, read as “0”
- Bit 6 **ADF**: A/D Converter Interrupt Request Flag
 0: No request
 1: Interrupt request
- Bit 5 **INT1F**: External Interrupt 1 Request Flag
 0: No request
 1: Interrupt request
- Bit 4 **INT0F**: External Interrupt 0 Request Flag
 0: No request
 1: Interrupt request
- Bit 3 **ADE**: A/D Converter Interrupt Control
 0: Disable
 1: Enable
- Bit 2 **INT1E**: External Interrupt 1 Control
 0: Disable
 1: Enable
- Bit 1 **INT0E**: External Interrupt 0 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	URF	MF2F	MF1F	MF0F	URE	MF2E	MF1E	MF0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7 **URF**: UART Interrupt Request Flag
0: No request
1: Interrupt request
- Bit 6 **MF2F**: Multi-function Interrupt 2 Request Flag
0: No request
1: Interrupt request
- Bit 5 **MF1F**: Multi-function Interrupt 1 Request Flag
0: No request
1: Interrupt request
- Bit 4 **MF0F**: Multi-function Interrupt 0 Request Flag
0: No request
1: Interrupt request
- Bit 3 **URE**: UART Interrupt Control
0: Disable
1: Enable
- Bit 2 **MF2E**: Multi-function Interrupt 2 Control
0: Disable
1: Enable
- Bit 1 **MF1E**: Multi-function Interrupt 1 Control
0: Disable
1: Enable
- Bit 0 **MF0E**: Multi-function Interrupt 0 Control
0: Disable
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TB1F	TB0F	SPIF	—	TB1E	TB0E	SPIE
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 **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 **SPIF**: SPI Interrupt Request Flag
0: No request
1: Interrupt request
- Bit 3 Unimplemented, read as “0”
- 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 **SPIE**: SPI Interrupt Control
0: Disable
1: Enable

• **MFIO Register**

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

Bit 7~6 Unimplemented, read as “0”

Bit 5 **CTM0AF**: CTM0 CCRA Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **CTM0PF**: CTM0 CCRP Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as “0”

Bit 1 **CTM0AE**: CTM0 CCRA Comparator Interrupt Control
0: Disable
1: Enable

Bit 0 **CTM0PE**: CTM0 CCRP Comparator Interrupt Control
0: Disable
1: Enable

• **MF1I Register**

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

Bit 7~6 Unimplemented, read as “0”

Bit 5 **CTM1AF**: CTM1 CCRA Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **CTM1PF**: CTM1 CCRP Comparator Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as “0”

Bit 1 **CTM1AE**: CTM1 CCRA Comparator Interrupt Control
0: Disable
1: Enable

Bit 0 **CTM1PE**: CTM1 CCRP Comparator Interrupt Control
0: Disable
1: Enable

• **MF12 Register**

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

- Bit 7~6 Unimplemented, read as “0”
- Bit 5 **DEF**: Data EEPROM Interrupt Request Flag
 0: No request
 1: Interrupt request
- Bit 4 **LVF**: LVD Interrupt Request Flag
 0: No request
 1: Interrupt request
- Bit 3~2 Unimplemented, read as “0”
- Bit 1 **DEE**: Data EEPROM Interrupt Control
 0: Disable
 1: Enable
- Bit 0 **LVE**: LVD Interrupt Control
 0: Disable
 1: Enable

Interrupt Operation

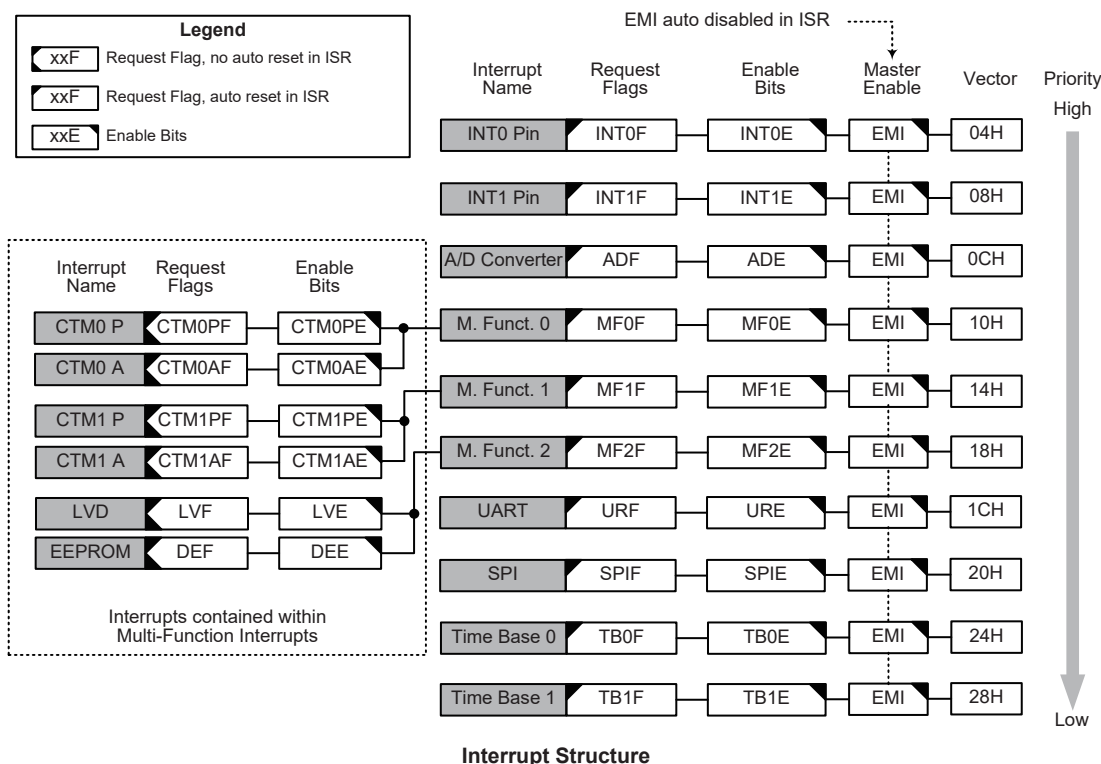
When the conditions for an interrupt event occur, such as a TM Comparator P, Comparator A match or A/D conversion completion etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition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

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

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

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

is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.



External Interrupts

The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register.

When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input. The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

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 Interrupt is contained within the Multi-function Interrupt. An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the respective EEPROM Interrupt vector will take place. When the EEPROM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

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.

Multi-function Interrupts

Within this device there are several Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupts, EEPROM interrupt and LVD interrupt.

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

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

UART Interrupt

Several individual UART conditions can generate a UART Interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. To allow the program to branch to the respective interrupt vector addresses, the global interrupt enable bit, EMI, and UART Interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of these conditions are created, a subroutine call to the UART Interrupt vector will take place. When the interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART section.

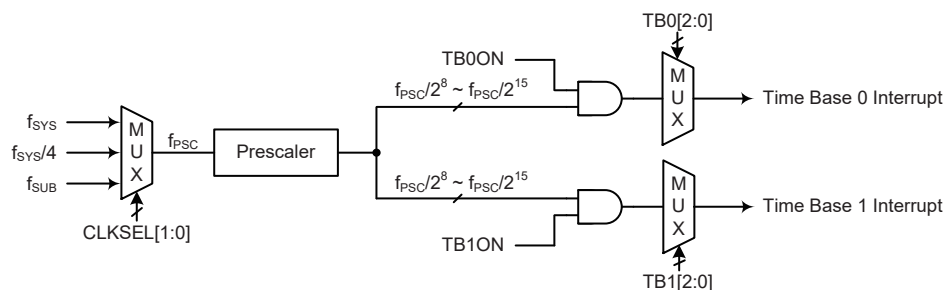
SPI Interrupt

The Serial Peripheral Interface Interrupt, also known as the SPI interrupt, will take place when the SPI Interrupt request flag, SPIF, is set, which occurs when a byte of data has been received or transmitted by the SPI interface or an SPI incomplete transfer occurs. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, SPIE, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the respective Interrupt vector, will take place. When the interrupt is serviced, the Serial Interface Interrupt flag, SPIF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Time Base Interrupts

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

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, f_{PSC} , originates from the internal clock source f_{SYS} , $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL1~CLKSEL0 bits in the PSCR register.



Time Base Interrupts

• **PSCR Register**

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

Bit 7~2 Unimplemented, read as “0”

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

• **TB0C Register**

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

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

Bit 6~3 Unimplemented, read as “0”

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

• **TB1C Register**

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

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

Bit 6~3 Unimplemented, read as “0”

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

Timer Module Interrupts

The Compact Type TMs each has two interrupts. All of the TM interrupts are contained within the Multi-function Interrupts. The Compact Type TM has two interrupt request flags of CTMnPF,

CTMnAF and two enable bits of CTMnPE, CTMnAE. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

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

Interrupt Wake-up Function

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

Programming Considerations

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

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

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

Every interrupt has the capability of waking up the microcontroller when it is in 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

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

LVD Register

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

• LVDC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	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 Detect
 1: Low Voltage Detect

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

Bit 3 **VBGEN**: Bandgap Buffer Control
 0: Disable
 1: Enable

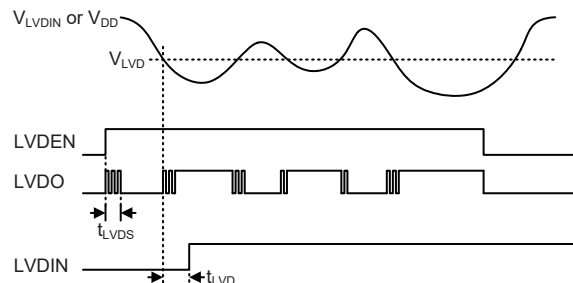
Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VBGEN bit is set to 1.

Bit 2~0 **VLVD2~VLVD0**: Select LVD Voltage
 000: $V_{LVDIN} \leq 1.04V$
 001: 2.2V
 010: 2.4V
 011: 2.7V
 100: 3.0V
 101: 3.3V
 110: 3.6V
 111: 4.0V

Note: When the VLVD bit field is set to 000B, the LVD function will be implemented by comparing an LVD reference voltage of 1.04V with the voltage derived from the LVDIN pin. Otherwise, the LVD function will operate by comparing the LVD reference voltage with a specific voltage value which is generated by the internal LVD circuit with V_{DD} when the VLVD bit field is set to any other value except 000B.

LVD Operation

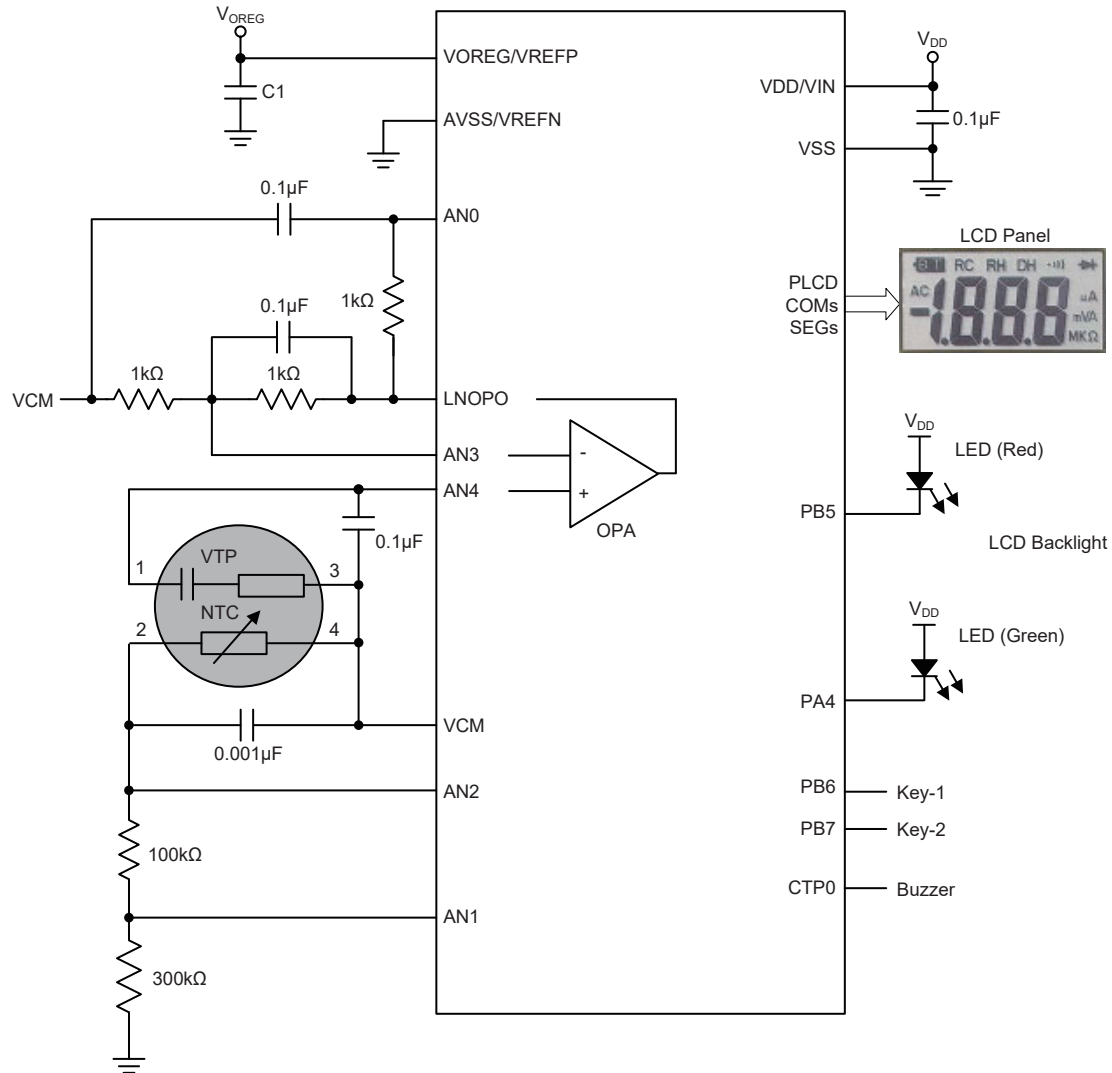
The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD} or the LVDIN pin input voltage with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.04V and 4.0V. When the power supply voltage, V_{DD} or the LVDIN pin input voltage fall below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. When the device is in the SLEEP mode, the low voltage detector will be disabled even if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage or the LVDIN pin input voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD} , there may be multiple bit LVDO transitions.



LVD Operation

The Low Voltage Detector interrupt is contained within the Multi-function interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. When the device is in the SLEEP mode, the low voltage detector will be disabled even 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} or LVDIN pin input voltage falls below the preset LVD voltage. This will cause the device to wake-up from the IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before the device enters the IDLE Mode. When LVD function is enabled, it is recommended to clear LVD flag first, and then enables interrupt function to avoid mistake action.

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 0.5 μ s and branch or call instructions would be implemented within 1 μ s. 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	1 ^{Note}	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	1 ^{Note}	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	1 ^{Note}	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	1 ^{Note}	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1 ^{Note}	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1 ^{Note}	Z
ORM A,[m]	Logical OR ACC to Data Memory	1 ^{Note}	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1 ^{Note}	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1 ^{Note}	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1 ^{Note}	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1 ^{Note}	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1 ^{Note}	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1 ^{Note}	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1 ^{Note}	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1 ^{Note}	C

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

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

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

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

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	2 ^{Note}	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	2 ^{Note}	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	2 ^{Note}	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	2 ^{Note}	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2 ^{Note}	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	2 ^{Note}	Z
LORM A,[m]	Logical OR ACC to Data Memory	2 ^{Note}	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2 ^{Note}	Z
LCPL [m]	Complement Data Memory	2 ^{Note}	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	2 ^{Note}	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2 ^{Note}	Z
Rotate			
LRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2 ^{Note}	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2 ^{Note}	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2 ^{Note}	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2 ^{Note}	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2 ^{Note}	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2 ^{Note}	None
LSET [m].i	Set bit of Data Memory	2 ^{Note}	None

Mnemonic	Description	Cycles	Flag Affected
Branch			
LSZ [m]	Skip if Data Memory is zero	2 ^{Note}	None
LSZA [m]	Skip if Data Memory is zero with data movement to ACC	2 ^{Note}	None
LSNZ [m]	Skip if Data Memory is not zero	2 ^{Note}	None
LSZ [m].i	Skip if bit i of Data Memory is zero	2 ^{Note}	None
LSNZ [m].i	Skip if bit i of Data Memory is not zero	2 ^{Note}	None
LSIZ [m]	Skip if increment Data Memory is zero	2 ^{Note}	None
LSDZ [m]	Skip if decrement Data Memory is zero	2 ^{Note}	None
LSIZA [m]	Skip if increment Data Memory is zero with result in ACC	2 ^{Note}	None
LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC	2 ^{Note}	None
Table Read			
LTABRD [m]	Read table to TBLH and Data Memory	3 ^{Note}	None
LTABRDL [m]	Read table (last page) to TBLH and Data Memory	3 ^{Note}	None
LITABRD [m]	Increment table pointer TBLP first and Read table to TBLH and Data Memory	3 ^{Note}	None
LITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	3 ^{Note}	None
Miscellaneous			
LCLR [m]	Clear Data Memory	2 ^{Note}	None
LSET [m]	Set Data Memory	2 ^{Note}	None
LSWAP [m]	Swap nibbles of Data Memory	2 ^{Note}	None
LSWAPA [m]	Swap nibbles of Data Memory with result in ACC	2	None

Note: 1. For these extended skip instructions, if the result of the comparison involves a skip then up to four cycles are required, if no skip takes place two cycles is required.

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

Instruction Definition

ADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC + [m] + C$
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	$[m] \leftarrow ACC + [m] + C$
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	$ACC \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC + x$
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	$[m] \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC \text{ "AND" } [m]$
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	$ACC \leftarrow ACC \text{ "AND" } x$
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	$[m] \leftarrow ACC \text{ "AND" } [m]$
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 \leftarrow Program Counter + 1 Program Counter \leftarrow addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] \leftarrow 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 \leftarrow 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 \leftarrow 0 PDF \leftarrow 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] \leftarrow $\overline{[m]}$
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 \leftarrow $\overline{[m]}$
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] \leftarrow ACC + 00H or [m] \leftarrow ACC + 06H or [m] \leftarrow ACC + 60H or [m] \leftarrow ACC + 66H
Affected flag(s)	C

DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	$[m] \leftarrow [m] - 1$
Affected flag(s)	Z
DECA [m]	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow [m] - 1$
Affected flag(s)	Z
HALT	Enter power down mode
Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.
Operation	$TO \leftarrow 0$ $PDF \leftarrow 1$
Affected flag(s)	TO, PDF
INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	$[m] \leftarrow [m] + 1$
Affected flag(s)	Z
INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow [m] + 1$
Affected flag(s)	Z
JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter \leftarrow addr
Affected flag(s)	None
MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	$ACC \leftarrow [m]$
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	$ACC \leftarrow x$
Affected flag(s)	None
MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	$[m] \leftarrow ACC$
Affected flag(s)	None

NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC \text{ "OR" } [m]$
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	$ACC \leftarrow ACC \text{ "OR" } x$
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	$[m] \leftarrow ACC \text{ "OR" } [m]$
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	$\text{Program Counter} \leftarrow \text{Stack}$
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	$\text{Program Counter} \leftarrow \text{Stack}$ $ACC \leftarrow x$
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	$\text{Program Counter} \leftarrow \text{Stack}$ $EMI \leftarrow 1$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C

RRCA [m]	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
SBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
SBC A, x	Subtract immediate data from ACC with Carry
Description	The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
SBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
SDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] - 1$ Skip if $[m]=0$
Affected flag(s)	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m] - 1$ Skip if $ACC=0$
Affected flag(s)	None

SET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	$[m] \leftarrow FFH$
Affected flag(s)	None
SET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	$[m].i \leftarrow 1$
Affected flag(s)	None
SIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] + 1$ 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	$ACC \leftarrow [m] + 1$ Skip if $ACC=0$
Affected flag(s)	None
SNZ [m].i	Skip if Data Memory is not 0
Description	If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if $[m].i \neq 0$
Affected flag(s)	None
SNZ [m]	Skip if Data Memory is not 0
Description	If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if $[m] \neq 0$
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	$ACC \leftarrow ACC - [m]$
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	$[m] \leftarrow ACC - [m]$
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	$ACC \leftarrow ACC - x$
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	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
Affected flag(s)	None
SZ [m]	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if $[m]=0$
Affected flag(s)	None
SZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m]$ Skip if $[m]=0$
Affected flag(s)	None
SZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if $[m].i=0$
Affected flag(s)	None

TABRD [m]	Read table (specific page) to TBLH and Data Memory
Description	The low byte of the program code (specific page) addressed by the table pointer pair (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
ITABRD [m]	Increment table pointer low byte first and read table to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
ITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "XOR" [m]
Affected flag(s)	Z
XORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC "XOR" [m]
Affected flag(s)	Z
XOR A,x	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "XOR" x
Affected flag(s)	Z

Extended Instruction Definition

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

LADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC + [m] + C$
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	$[m] \leftarrow ACC + [m] + C$
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	$ACC \leftarrow ACC + [m]$
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	$[m] \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC \text{ "AND" } [m]$
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	$[m] \leftarrow ACC \text{ "AND" } [m]$
Affected flag(s)	Z
LCLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	$[m] \leftarrow 00H$
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	$[m].i \leftarrow 0$
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	$ACC \leftarrow [m]$
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	$[m] \leftarrow ACC$
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	$ACC \leftarrow ACC \text{ "OR" } [m]$
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	$[m] \leftarrow ACC \text{ "OR" } [m]$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC \leftarrow ACC - [m] - \overline{C}$
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	$[m] \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

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

LSNZ [m]	Skip if Data Memory is not 0
Description	If the content of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m] \neq 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	$ACC \leftarrow ACC - [m]$
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	$[m] \leftarrow ACC - [m]$
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	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
Affected flag(s)	None
LSZ [m]	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m]=0
Affected flag(s)	None
LSZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m]$ Skip if [m]=0
Affected flag(s)	None

LSZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if [m].i=0
Affected flag(s)	None
LTABRD [m]	Read table (current page) to TBLH and Data Memory
Description	The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
LTABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
LITABRD [m]	Increment table pointer low byte first and read table to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
LITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
LXOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "XOR" [m]
Affected flag(s)	Z
LXORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC "XOR" [m]
Affected flag(s)	Z

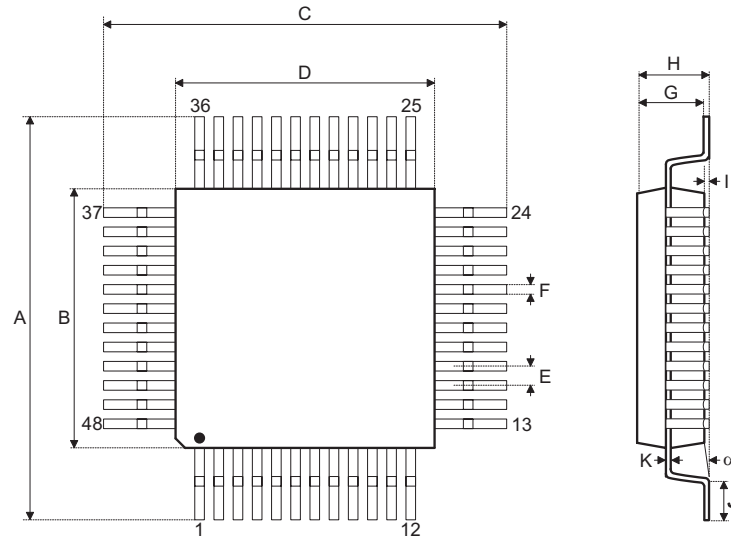
Package Information

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

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

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- Carton information

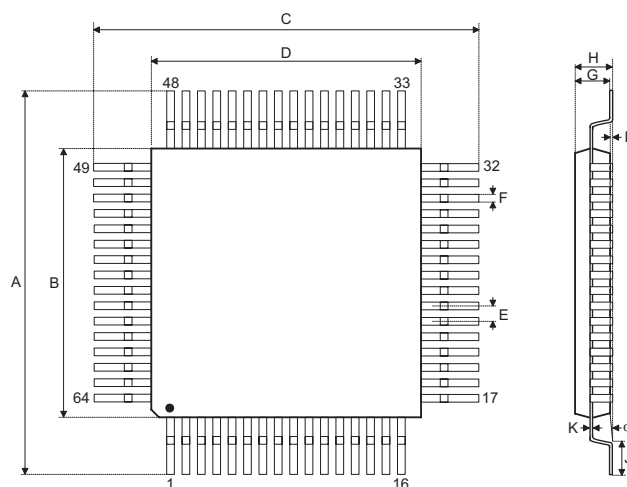
48-pin LQFP (7mm×7mm) Outline Dimensions



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°

Copyright© 2019 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at <http://www.holtek.com>.