Ultra-Low Power Flash MCU with LCD & EEPROM

HT69F2562

Revision: V1.00 Date: January 26, 2018

www.holtek.com

Table of Contents

Features

CPU Features

- Operating Voltage
  - f_{SYS}=4MHz: 1.8V~5.5V
  - f_{SYS}=8MHz: 2.2V~5.5V
  - f_{SYS}=12MHz: 2.7V~5.5V
- Up to 0.33μs instruction cycle with 12MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillators
  - Internal High Speed 4/8/12MHz RC Oscillator – HIRC
  - External Low Speed 32.768kHz Crystal – LXT
  - Internal Low Speed 32kHz RC Oscillator– LIRC, for power on reset and LVD/LVR functions only
- 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
- 16-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 16K×16
- RAM Data Memory: 2304×8
- True EEPROM Memory: 128×8
- Watchdog Timer function
- 20 bidirectional I/O lines
- Four external interrupt lines shared with I/O pins
- Multiple Timer Modules for time measurement, input capture, compare match output, PWM output function or single pulse output function
- Universial Serial Interface Module – USIM for SPI, I^2C or UART communication
- Single serial SPI interface – SPIA
- Dual Time-Base functions for generation of fixed time interrupt signals
- LCD driver function
  - SEGs × COMs: 32×4
  - Duty type: 1/4 duty
  - Bias level: 1/3 bias
  - Bias type: C type
  - Waveform type: type A or type B
- Low voltage reset function – LVR
- Low voltage detect function – LVD
- Package type: 64-pin LQFP

General Description

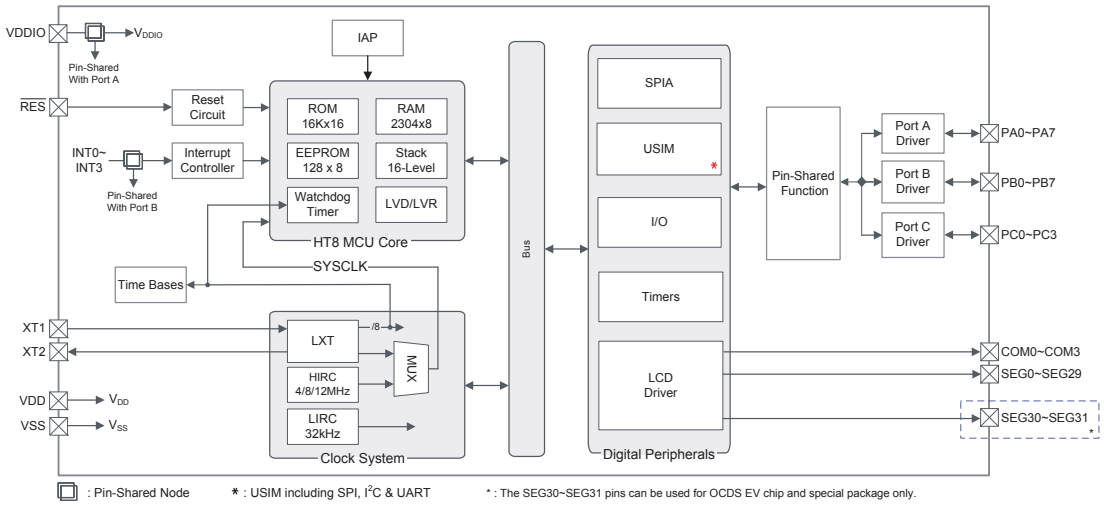
The device is a Flash Memory LCD 8-bit high performance RISC architecture microcontrollers, designed for applications that LCD display products. Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and features. Other memory includes 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.

Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI, I^2C and UART interface functions, these popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

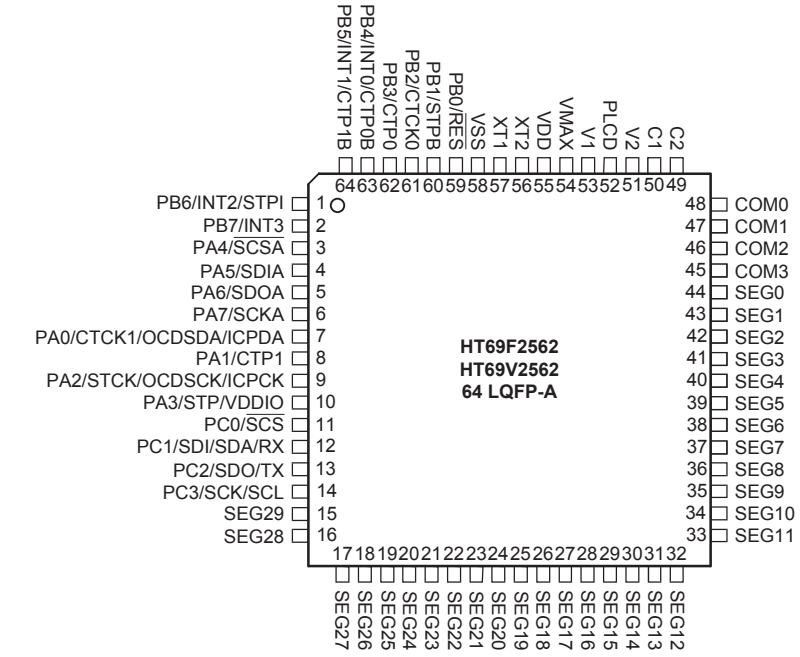
A full choice of external low and internal high oscillator functions are provided including fully integrated system oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources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

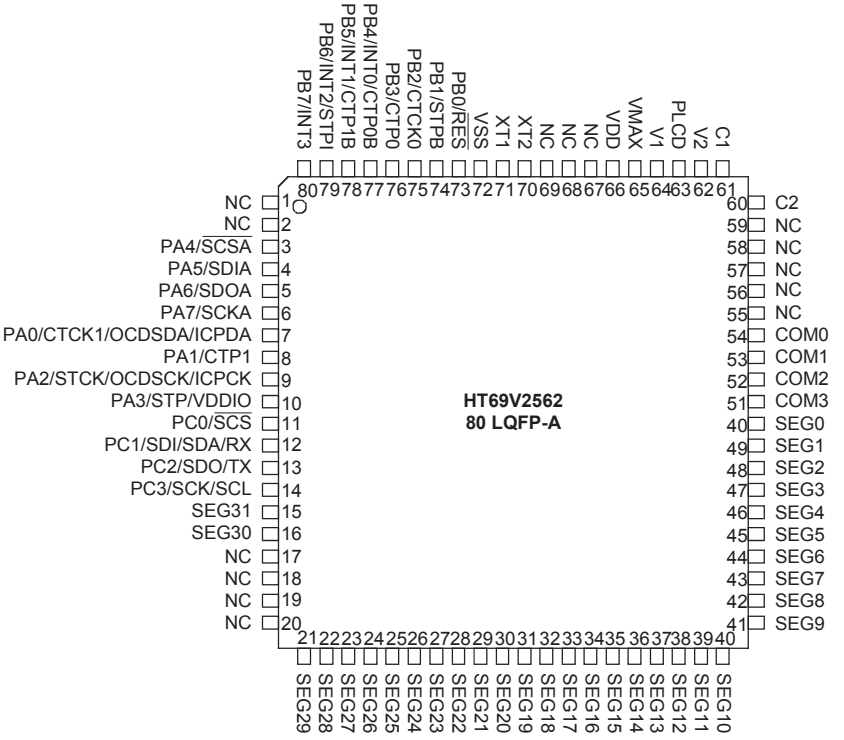
The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the device will find excellent use in applications such as Visible Rewrite Cards and Electronic Tags in addition to many others.

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 OCDSDA and OCDSCK pins are supplied as OCDS dedicated pins and as such only available for the HT69V2562 device which is the OCDS EV chip for the HT69F2562 device.

Pin Description

With the exception of the power pins and some relavant transformer control pins, all pins on the device can be referenced by their Port names, e.g. PA0, PA1 etc, which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Timer Module pins etc. The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/CTCK1/ OCDSDA/ ICPDA	PA0	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	CTCK1	—	ST	—	CTM1 clock input
	OCDSDA	—	ST	CMOS	OCDS Data/Address, for EV chip only
	ICPDA	—	ST	CMOS	ICP Data/Address
PA1/CTP1	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	CTP1	PAS0	—	CMOS	CTM1 output
PA2/STCK/ OCDSCK/ ICPCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	STCK	—	ST	—	STM clock input pin
	OCDSCK	—	ST	—	OCDS Clock pin, for EV chip only
	ICPCK	—	ST	—	ICP Clock pin
PA3/STP/ VDDIO	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	STP	PAS0	—	CMOS	STM output
	VDDIO	PAS0 PMPS	PWR	—	Power supply for SPI/I²C/UART input/output pins
PA4/$\overline{SCSA}$	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{SCSA}$	PAS1	ST	CMOS	SPIA slave select pin
PA5/SDIA	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDIA	PAS1	ST	—	SPIA serial data input
PA6/SDOA	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDOA	PAS1	—	CMOS	SPIA serial data output
PA7/SCKA	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCKA	PAS1	ST	CMOS	SPIA serial clock
PB0/$\overline{RES}$	PB0	PBPU RSTC	ST	CMOS	General purpose I/O. Register enabled pull-high.
	$\overline{RES}$	RSTC	ST	—	External reset input
PB1/STPB	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	STPB	PBS0	—	CMOS	STM inverting output
PB2/CTCK0	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	CTCK0	—	ST	—	CTM0 clock input

Pin Name	Function	OPT	I/T	O/T	Description
PB3/CTP0	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	CTP0	PBS0	—	CMOS	CTM0 output
PB4/INT0/ CTP0B	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high.
	INT0	PBS1 INTEG INTC0	ST	—	External interrupt input 0
	CTP0B	PBS1	—	CMOS	CTM0 inverting output
PB5/INT1/ CTP1B	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high.
	INT1	PBS1	ST	—	External interrupt input 1
	CTP1B	PBS1	—	CMOS	CTM1 inverting output
PB6/INT2/STPI	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	INT2	INTEG INTC2	ST	—	External interrupt input 2
	STPI	—	ST	—	STM capture input
PB7/INT3	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	INT3	INTEG INTC2	ST	—	External interrupt input 3
PC0/$\overline{SCS}$	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	$\overline{SCS}$	PCS0	ST	CMOS	SPI slave select pin
PC1/SDI/SDA/ RX	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	SDI	PCS0	ST	—	SPI serial data input
	SDA	PCS0	ST	NMOS	I^2C data line
	RX	PCS0	ST	—	UART RX serial data input
PC2/SDO/TX	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	SDO	PCS0	—	CMOS	SPI serial data output
	TX	PCS0	—	CMOS	UART TX serial data output
PC3/SCK/SCL	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high.
	SCK	PCS0	ST	CMOS	SPI serial clock
	SCL	PCS0	ST	NMOS	I^2C clock line
XT2	XT2	—	—	LXT	LXT oscillator pin
XT1	XT1	—	LXT	—	LXT oscillator pin
LCD					
SEG0~SEG29	SEG0~29	—	—	LCD	LCD Segment output
SEG30~SEG31	SEG30~31	—	—	LCD	LCD Segment output, can be used for OCDS EV chip and special package only.
COM0~COM3	COM0~3	—	—	LCD	LCD Common output
VMAX	VMAX	—	PWR	—	LCD maximum voltage, connect to V_{DD} or V1
PLCD	PLCD	—	PWR	AN	LCD power supply
V1	V1	—	PWR	AN	LCD power supply
V2	V2	—	PWR	AN	LCD power supply
C1	C1	—	—	AN	LCD voltage pump
C2	C2	—	—	AN	LCD voltage pump

Pin Name	Function	OPT	I/T	O/T	Description
Power					
VDD	VDD	—	PWR	—	Digital positive power supply
VSS	VSS	—	PWR	—	Digital negative power supply

Legend: I/T: Input type
OPT: Optional by register option
CMOS: CMOS output
AN: Analog signal
LXT: Low frequency crystal oscillator
O/T: Output type
ST: Schmitt Trigger input
NMOS: NMOS output
PWR: Power
LCD: LCD COM/SEG output

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}-0.3V to V_{SS}+6.0V

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

Storage Temperature ..... -50°C to 125°C

Operating Temperature ..... -40°C to 70°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

Ta=-40°C~70°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	f_{SYS}=4MHz	1.8	—	5.5	V
		f_{SYS}=8MHz	2.2	—	5.5	
		f_{SYS}=12MHz	2.7	—	5.5	
	Operating Voltage – LXT	f_{SYS}=32768Hz	1.8	—	5.5	V

Standby Current Characteristics

Ta=25°C

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max.	Unit
		V_{DD}	Conditions				70°C	
I_{STB}	SLEEP Mode	1.8V	WDT off, Time Base off, LCD off (LXT on)	—	0.10	0.15	0.70	μA
		3V		—	0.12	0.15	1.00	
		5V		—	0.20	0.50	1.20	
		1.8V	WDT off, Time Base on, LCD off (LXT on)	—	0.12	0.22	0.70	
		3V		—	0.15	0.22	1.00	
		5V		—	0.23	0.60	1.20	
		1.8V	WDT on, Time Base on, LCD off (LXT on)	—	0.15	0.22	1.00	
		3V		—	0.18	0.22	1.50	
		5V		—	0.30	0.75	1.80	
	IDLE0 Mode – LXT	1.8V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3.0	5.0	6.0	
		5V		—	5.0	10	12	
	IDLE1 Mode – HIRC	1.8V	f_{SUB} on, f_{SYS}=4MHz	—	0.144	0.200	0.240	mA
		3V		—	0.180	0.250	0.300	
		5V		—	0.400	0.600	0.720	
		2.2V	f_{SUB} on, f_{SYS}=8MHz	—	0.3	0.6	0.8	
		3V		—	0.5	1.0	1.8	
		5V		—	1.0	2.0	2.2	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	0.4	0.8	1.0	
		3V		—	0.6	1.2	1.4	
		5V		—	1.2	2.4	2.6	

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

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

Operating Current Characteristics

Ta=25°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LXT	1.8V	f_{SYS}=32768Hz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	1.8V	f_{SYS}=4MHz	—	0.3	0.5	mA
		3V		—	0.4	0.6	
		5V		—	0.8	1.2	
		2.2V	f_{SYS}=8MHz	—	0.8	1.2	
		3V		—	1.0	1.5	
		5V		—	2.0	3.0	
		2.7V	f_{SYS}=12MHz	—	1.2	2.2	
		3V		—	1.5	2.7	
		5V		—	3.0	4.5	

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

A.C. Characteristics

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

High Speed Internal Oscillator – HIRC – Frequency Accuracy

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

4/8/12MHz

Symbol	Parameter	Test Conditions		Min	Typ	Max	Unit
		V_{DD}	Temp.				
f_{HIRC}	4MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	4	+1%	MHz
			-40°C~70°C	-2%	4	+2%	
		2.2V~5.5V	25°C	-2.5%	4	+2.5%	
			-40°C~70°C	-3%	4	+3%	
		1.8V~5.5V	25°C	-4%	4	+4%	
			-40°C~70°C	-5%	4	+5%	
	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~70°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~70°C	-3%	8	-3%	
	12MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	12	+1%	MHz
			-40°C~70°C	-2%	12	+2%	
		2.7V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C~70°C	-3%	12	+3%	

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

Low Speed Internal Oscillator Characteristics – LIRC

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	2.2V~5.5V	25°C	-5%	32	+5%	kHz
			-40°C~70°C	-10%	32	+10%	
t_{START}	LIRC Start Up Time	—	—	—	—	100	μs

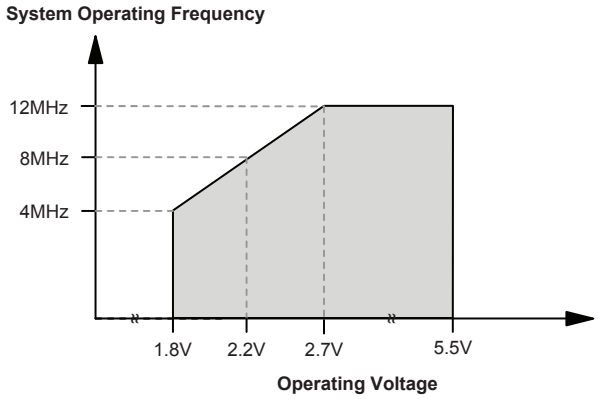
Low Speed Crystal Oscillator Characteristics – LXT

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{LXT}	Oscillator Frequency	1.8V~5.5V	—	—	32768	—	Hz
Duty Cycle	Duty Cycle	—	—	48	50	52	%
t_{START}	Start Up Time	3V/5V	—	—	—	600	ms
R_{NEG}	Negative Resistance	1.8V	—	3×ESR	—	—	Ω

*: C1 and C2 are external components. C1=C2=7pF. C_L<7pF, ESR=65kΩ (Max.).

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~70°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time Wake-up from Condition where f_{SYS} is Off	—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LXT}$	—	1024	—	t_{SYS}
	System Start-up Time Wake-up from Condition where f_{SYS} is On	—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LXT}$	—	2	—	t_{SYS}
	System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode	—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
		—	f_{LXT} switches from off → on	—	1024	—	t_{LXT}
t_{RSTD}	System Reset Delay Time Reset Source from Power-on Reset or LVR Hardware Reset	—	RR_{POR}=5 V/ms	42	48	54	ms
	System Reset Delay Time LVRC/WDTC/RSTC Software Reset	—	—				
	System Reset Delay Time Reset Source from WDT Overflow or Reset Pin Reset	—	—	14	16	18	ms
t_{SRESET}	Minimum Software Reset Pulse 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. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output D.C. Characteristics

Ta=25°C

Input/Output (without Multi-power) D.C. Characteristics

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports except PC0~PC3 Pins	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports except PC0~PC3 Pins	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Pins except PC0~PC3 Pins	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Pins except PC0~PC3 Pins	3V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	
R_{PH}	Pull-high Resistance for I/O Ports(Note)	3V	LVPU=0 PxPU=FFH (Px: PA, PB, PC)	20	60	100	kΩ
		5V		10	30	50	
		3V	LVPU=1 PxPU=FFH (Px: PA, PB, PC)	6.67	15	23	
		5V		3.5	7.5	12	
I_{LEAK}	Input Leakage Current except PC0~PC3 Pins	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{TPI}	TM Capture Input Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TCK}	TM Clock Input Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	Interrupt Input Pin Minimum Pulse Width	—	—	10	—	—	μs
t_{RES}	External Reset Minimum Pulse Width	—	—	10	—	—	μs

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	VDD Power Supply for PC0~PC3 Pins	—	—	1.8	5.0	5.5	V
V_{DDIO}	VDDIO Power Supply for PC0~PC3 Pins	—	—	1.8	—	V_{DD}	V
V_{IL}	Input Low Voltage for PC0~PC3 Pins	5V	Pin power=V_{DD} or V_{DDIO}, $V_{DDIO}=V_{DD}$	0	—	1.5	V
		—	Pin power=V_{DD} or V_{DDIO}	0	—	0.2 (V_{DD}/ V_{DDIO})	
V_{IH}	Input High Voltage for PC0~PC3 Pins	5V	Pin power=V_{DD} or V_{DDIO}, $V_{DDIO}=V_{DD}$	3.5	—	5.0	V
		—	Pin power=V_{DD} or V_{DDIO}	$0.8V_{DD}$	—	V_{DD}/ V_{DDIO}	
I_{OL}	Sink Current for I/O Pins	3V	$V_{OL}=0.1(V_{DD}/V_{DDIO})$, $V_{DDIO}=V_{DD}$	16	32	—	mA
		5V	$V_{OL}=0.1(V_{DD}/V_{DDIO})$, $V_{DDIO}=V_{DD}$	32	65	—	
			$V_{OL}=0.1V_{DDIO}$, $V_{DDIO}=3V$	20	40	—	
I_{OH}	Source Current for I/O Pins	3V	$V_{OH}=0.9(V_{DD}/V_{DDIO})$, $V_{DDIO}=V_{DD}$	-4	-8	—	mA
		5V	$V_{OH}=0.9(V_{DD}/V_{DDIO})$, $V_{DDIO}=V_{DD}$	-8	-16	—	
			$V_{OH}=0.9V_{DDIO}$, $V_{DDIO}=3V$	-2.5	-5.0	—	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
R_{PH}	Pull-high Resistance for PC0~PC3 Pins (Note)	3V	$V_{DDIO}=V_{DD}$, LVPU=0 PxPU=FFH (Px: PA, PB, PC)	20	60	100	kΩ
		5V	$V_{DDIO}=V_{DD}$, LVPU=0 PxPU=FFH (Px: PA, PB, PC)	10	30	50	kΩ
			V_{DDIO}=3V, LVPU=0 PxPU=FFH (Px: PA, PB, PC)	36	110	180	
		3V	$V_{DDIO}=V_{DD}$, LVPU=1 PxPU=FFH (Px: PA, PB, PC)	6.67	15	23	kΩ
		5V	$V_{DDIO}=V_{DD}$, LVPU=1 PxPU=FFH (Px: PA, PB, PC)	3.5	7.5	12	kΩ
			V_{DDIO}=3V, LVPU=1 PxPU=FFH (Px: PA, PB, PC)	9.0	27.5	45	
I_{LEAK}	Input Leakage Current for PC0~PC3 Pins	5V	$V_{IN}=V_{SS}$ or $V_{IN}=V_{DD}$ or V_{DDIO}	—	—	±1	μA

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

Memory Electrical Characteristics

Ta=-40°C~70°C

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	
I_{DDPGM}	Programming / Erase Current on V_{DD}	—	—	—	—	5.0	mA
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	–	Device in SLEEP Mode	1.0	–	–	V

LVD/LVR Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 1.70V	-5%	1.70	+5%	V
			LVR enable, voltage select 1.90V		1.90		
			LVR enable, voltage select 2.55V	-3%	2.55	+3%	
			LVR enable, voltage select 3.15V		3.15		
			LVR enable, voltage select 3.80V		3.80		
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 1.80V	-5%	1.80	+5%	V
			LVD enable, voltage select 1.90V		1.90		
			LVD enable, voltage select 2.00V		2.00		
			LVD enable, voltage select 2.10V		2.10		
			LVD enable, voltage select 2.20V		2.20		
			LVD enable, voltage select 2.30V		2.30		
			LVD enable, voltage select 2.40V		2.40		
			LVD enable, voltage select 2.50V		2.50		
			LVD enable, voltage select 2.60V		2.60		
			LVD enable, voltage select 2.70V		2.70		
			LVD enable, voltage select 2.80V		2.80		
			LVD enable, voltage select 2.90V		2.90		
			LVD enable, voltage select 3.00V		3.00		
			LVD enable, voltage select 3.30V		3.30		
			LVD enable, voltage select 3.60V		3.60		
			LVD enable, voltage select 4.00V		4.00		
I_{LVRLVD}	Operating Current	3V	LVD enable, LVR enable, V_{LVR}=1.9V, V_{LVD}=2V	—	—	10	μA
		5V		—	8	15	
t_{LVDS}	LVDO Stable Time	—	For LVR enable, LVD off → on	—	—	15	μs
		—	For LVR disable, LVD off → on	—	—	150	
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs
I_{LVR}	Additional Current for LVR Enable	—	LVD disable	—	—	11	μA
I_{LVD}	Additional Current for LVD Enable	—	LVR disable	—	—	11	μA

LCD Electrical Characteristics

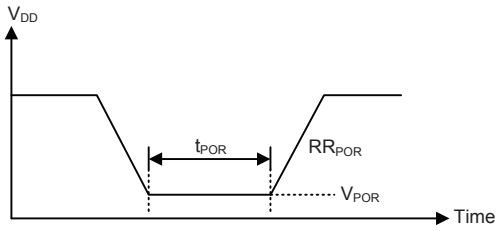
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IN}	LCD Operating Voltage	—	Power supply from PLCD pin	2.0	—	3.7	V
		—	Power supply from V1 pin	3.0	—	5.5	
		—	Power supply from V2 pin	1.0	—	1.8	
		—	Power supply from V_A	3.0	—	5.5	
		—	Power supply from V_B	2.0	—	3.7	
		—	Power supply from V_C	2.2	—	5.5	
I_{LCD}	Additional Current for LCD Enable (C type)	3V	No load, V_A=V1=V_{DD}, 1/3 Bias LCDP[1:0]=11B, LCDPCK[2:0]=000B	—	0.3	0.6	μA
		5V		—	0.5	1.0	
		3V	No load, V_A=V1=V_{DD}, 1/3 Bias LCDP[1:0]=01B (V_C=DPN Vref) LCDPCK[2:0]=000B	—	1.6	3	μA
		5V		—	1.8	5	
I_{LCDOL}	LCD Common and Segment Sink Current	3V	V_{OL}=0.1V_A	210	420	—	μA
		5V		350	700	—	
I_{LCDOH}	LCD Common and Segment Source Current	3V	V_{OH}=0.9V_A	-80	-160	—	μA
		5V		-180	-360	—	

Power-on Reset Characteristics

Ta=25°C

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



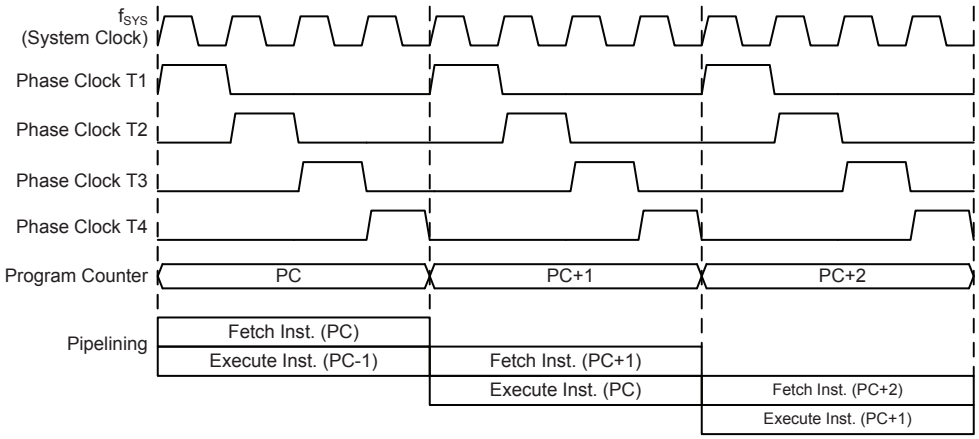
System Architecture

A key factor in the high-performance features of the range of microcontrollers is attributed to their internal system architecture. The 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 control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

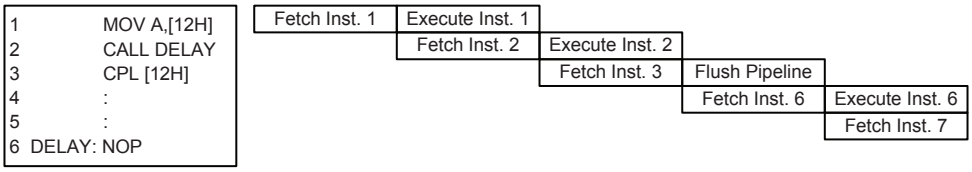
Clocking and Pipelining

The main system clock, derived from either an LXT or HIRC 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. For the device whose memory capacity is greater than 8K words the Program Memory address may be located in a certain program memory bank which is selected by the program memory bank pointer bit PBP0. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

Program Counter	
Program Counter High Byte	**PCL Register**
PBP0, PC12~PC8	PCL7~PCL0

Program Counter

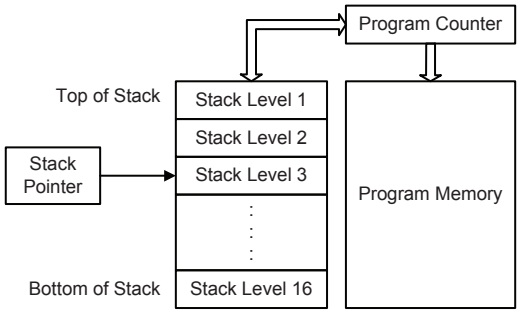
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly. However, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organised into 16 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

Counter save in the stack will be lost.



Arithmetic and Logic Unit – ALU

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

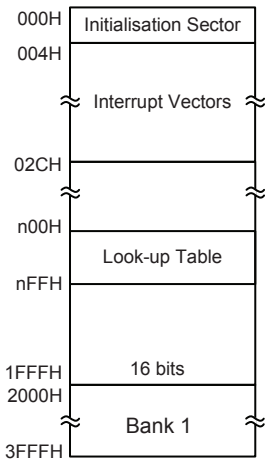
- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
  LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,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

Flash Program Memory

The Program Memory is the location where the user code or program is stored. For 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 16K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

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

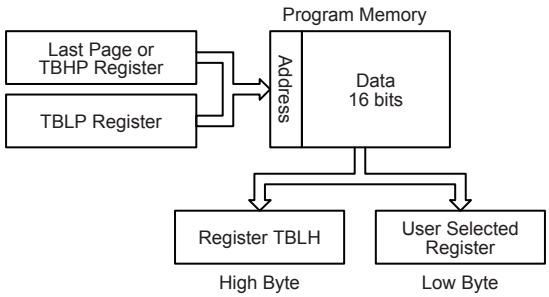


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "1F00H" which is located in the Bank 1 and refers to the start address of the last page within the 16K 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 "3F06H" 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

```
rombank 1 code1
ds .section 'data'
tempreg1 db ?   ; temporary register #1
tempreg2 db ?   ; temporary register #2
:
:
code0 .section 'code'
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,3Fh       ; initialise high table pointer
mov tbhp,a      ; It is not necessary to set tbhp register if executing "tabrdl"
                ; instruction
:
:
tabrd tempreg1  ; transfers value in table referenced by table pointer data at program
                ; memory address "3F06H" transferred to tempreg1 and TBLH
dec tblp        ; reduce value of table pointer by one
tabrd tempreg2  ; transfers value in table referenced by table pointer
                ; data at program memory address "3F05H" transferred to
                ; tempreg2 and TBLH in this example the data "1AH" is
                ; transferred to tempreg1 and data "0FH" to register tempreg2
:
:
code1 .section 'code'
org 1F00h       ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
```

In Circuit Programming – ICP

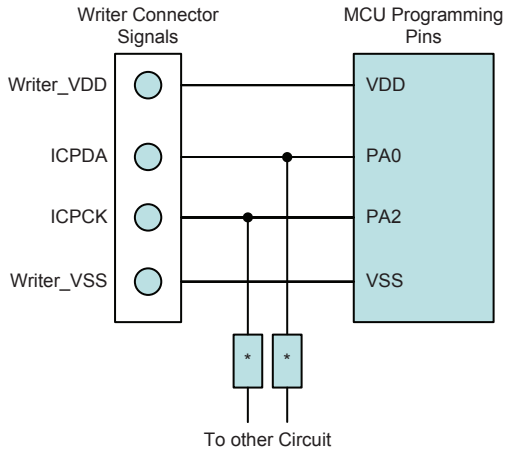
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, a means of programming the microcontroller in-circuit has provided using a 4-pin interface. This provides manufacturers 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 and EEPROM data Memory can both 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 OCDSCK pins to the HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK 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.

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

In Application Programming – IAP

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

Flash Memory Read/Write Size

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

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

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

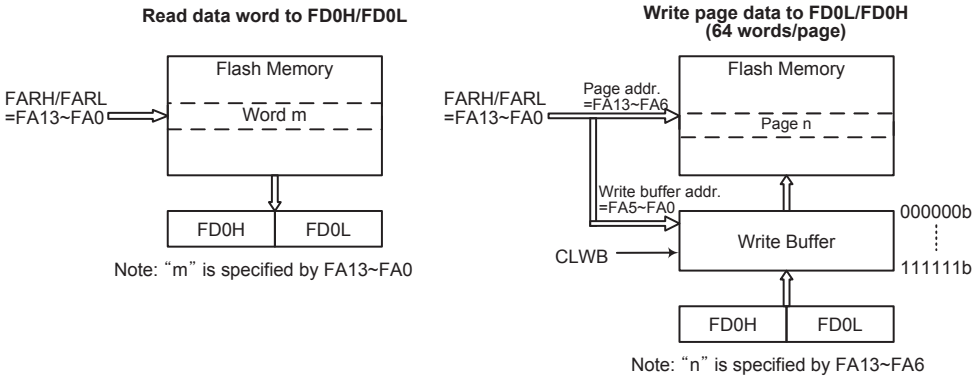
Operations	Format
Erase	64 words/page
Write	64 words/time
Read	1 word/time
Note: Page size=Write buffer size=64 words.	

IAP Read/Write Format

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

"x": don't care

Erase Page Number and Selection



Flash Memory IAP Read/Write Structure

Write Buffer

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

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

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

IAP Flash Program Memory Registers

There are two address registers, four 16-bit data registers and three control registers. The address and data register pairs are located in Sector 0 while the control registers are located in Sector 1. Read and Write operations to the Flash memory are carried out using 16-bit data operations using the address and data registers and the control register. Several registers control the overall operation of the internal Flash Program Memory. The address registers are named FARL and FARH, the data registers are named FDnL and FDnH and the control registers are named FC0, FC1 and FC2. As the address and data register pairs are located in Sector 0, they can be directly accessed in the same way as any other Special Function Register. The control registers, being located in Sector 1, can be addressed directly only using the corresponding extended instructions or can be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pairs and Indirect Addressing Register, IAR1 or IAR2.

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

IAP Register List

- **FARL Register**

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

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

- **FARH Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 Flash Memory Address bit 13 ~ bit 8

- **FD0L Register**

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

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

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

- **FD0H Register**

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

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

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

- **FD1L Register**

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

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

- **FD1H Register**

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

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

- **FD2L Register**

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

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

- **FD2H Register**

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

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

- **FD3L Register**

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

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

- **FD3H Register**

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

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

- **FC0 Register**

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

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

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

Bit 3 **FWPEN**: Flash memory Erase/Write function enable procedure Trigger
0: Erase/Write function enable procedure is not triggered or procedure timer times out
1: Erase/Write function enable procedure is triggered and procedure timer starts to count
This bit is used to activate the flash memory Erase/Write function enable procedure and an internal timer. It is set by the application programs and then cleared by the hardware when the internal timer times out. The correct patterns must be written into the FD1L/FD1H, FD2L/FD2H and FD3L/FD3H register pairs respectively as soon as possible after the FWPEN bit is set high.

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

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

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

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

- **FC1 Register**

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

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

- **FC2 Register**

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

Bit 7~1 Unimplemented, read as "0"

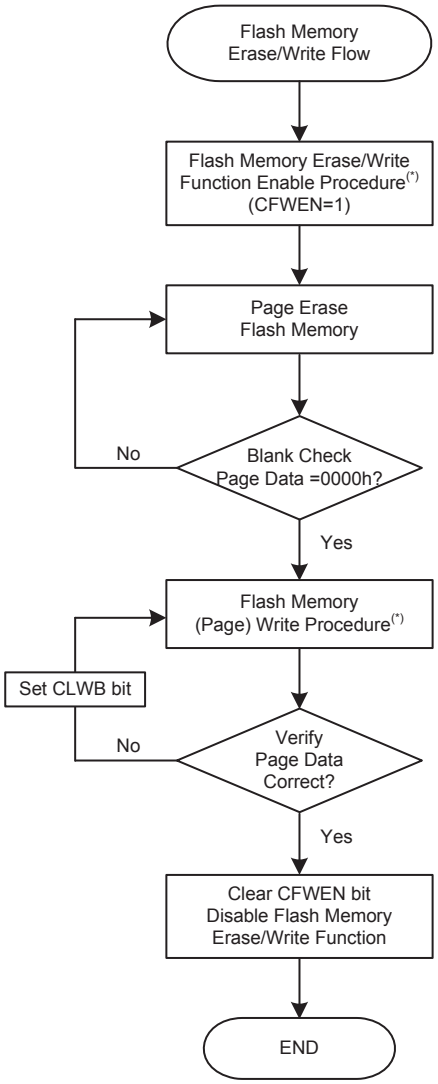
Bit 0 **CLWB**: Flash memory Write Buffer Clear control
0: Do not initiate a Write Buffer Clear process or indicating that a Write Buffer Clear process has completed
1: Initiate Write Buffer Clear process
This bit is set by software and cleared by hardware when the Write Buffer Clear process has completed.

Flash Memory Erase/Write Flow

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

Flash Memory Erase/Write Flow Descriptions:

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



Flash Memory Erase/Write Flow

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

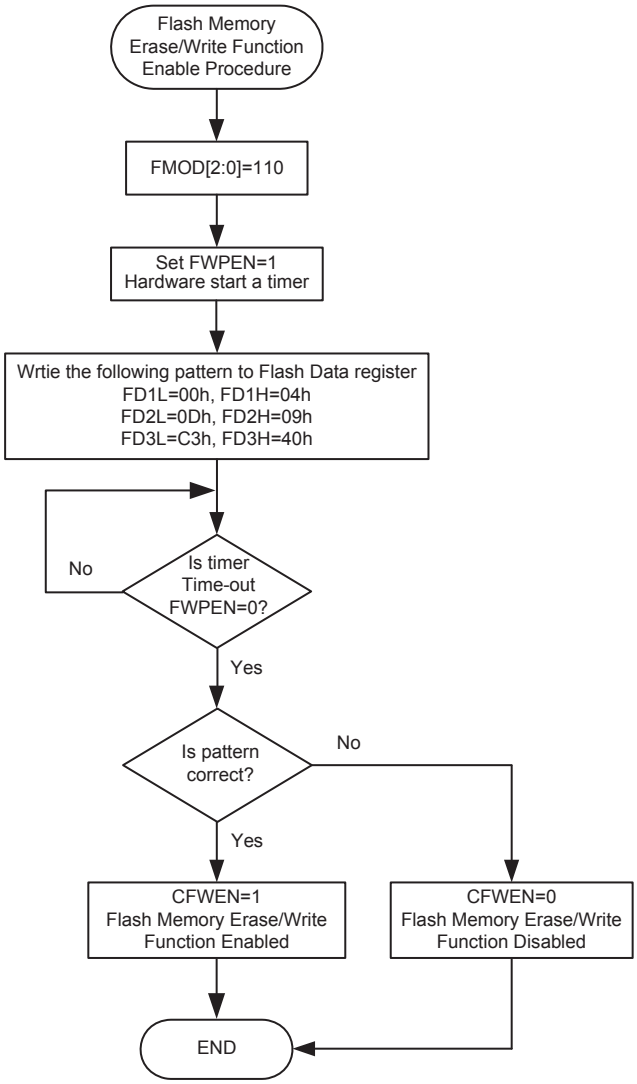
Flash Memory Erase/Write Function Enable Procedure

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

Flash Memory Erase/Write Function Enable Procedure Description

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

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



Flash Memory Erase/Write Function Enable Procedure

Flash Memory Write Procedure

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

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

Flash Memory Consecutive Write Description

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

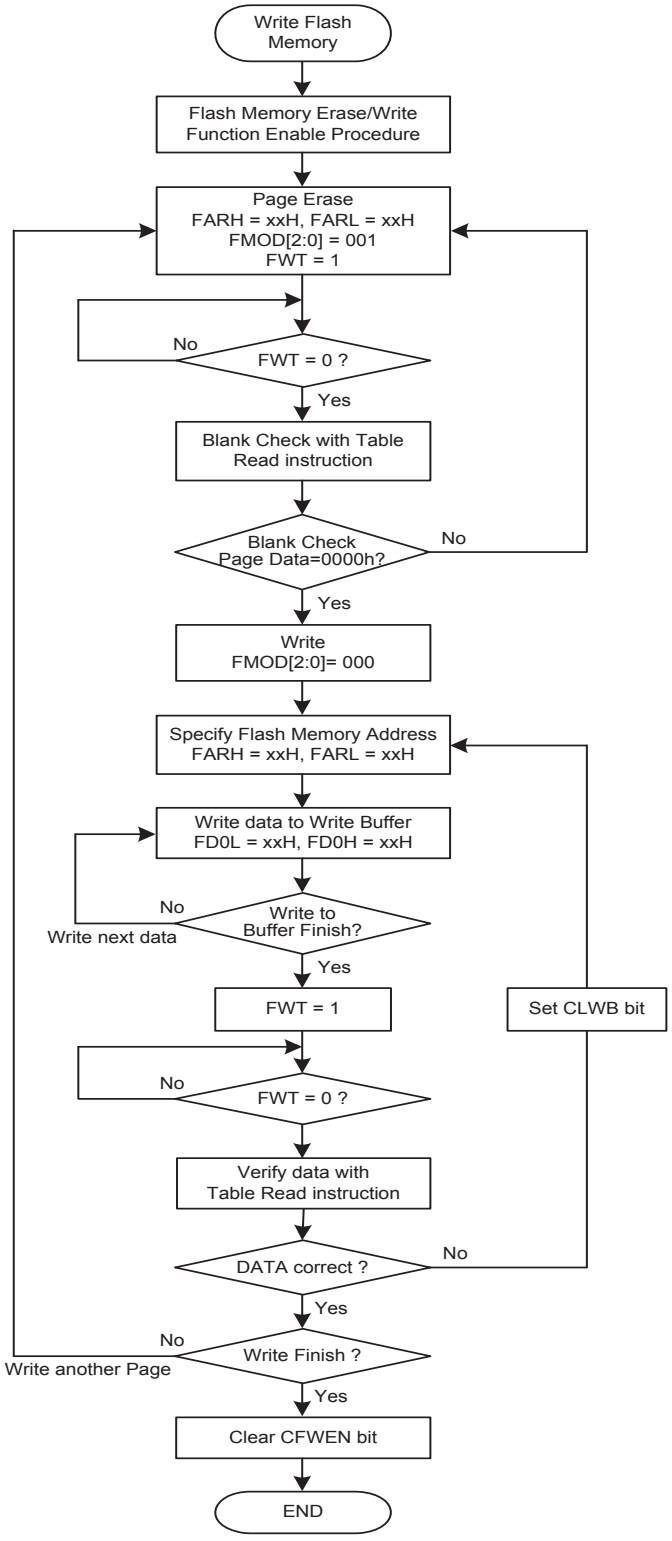
1. Activate the "Flash Memory Erase/Write function enable procedure". Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the "Flash Memory Erase/Write function enable procedure" for more details.
2. Set the FMOD field to "001" to select the erase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

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

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

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

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



Flash Memory Consecutive Write Procedure

Note: 1. When the FWT bit is set high all CPU operations will temporarily cease.

2. It will take a typical time of 2.2ms for the FWT bit state changing from high to low.

Flash Memory Non-Consecutive Write Description

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

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

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

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

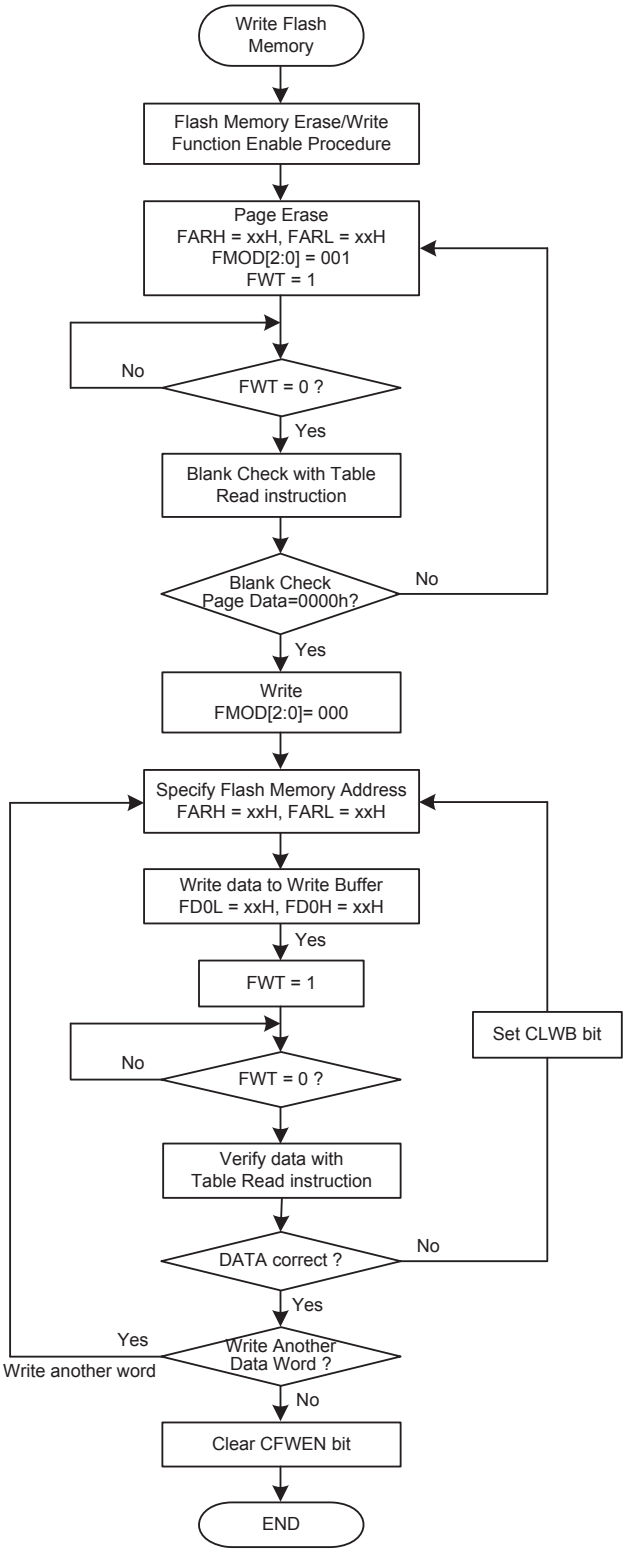
   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to "000" to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FRARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
6. Set the FWT bit high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

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

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

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

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



Flash Memory Non-Consecutive Write Procedure

Note: 1. When the FWT bit is set high all CPU operations will temporarily cease.

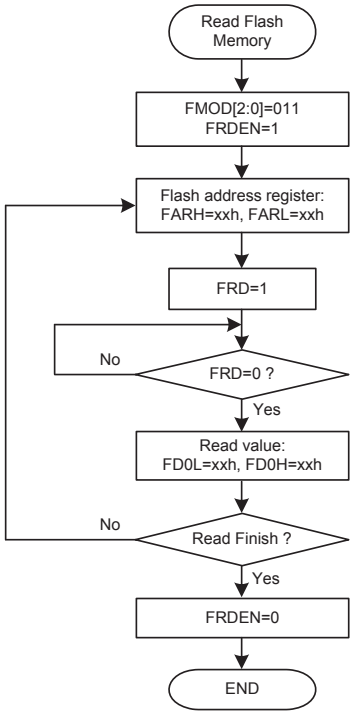
2. It will take a typical time of 2.2ms for the FWT bit state changing from high to low.

Important Points to Note for Flash Memory Write Operations

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

Flash Memory Read Procedure

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



Flash Memory Read Procedure

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

Data Memory

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

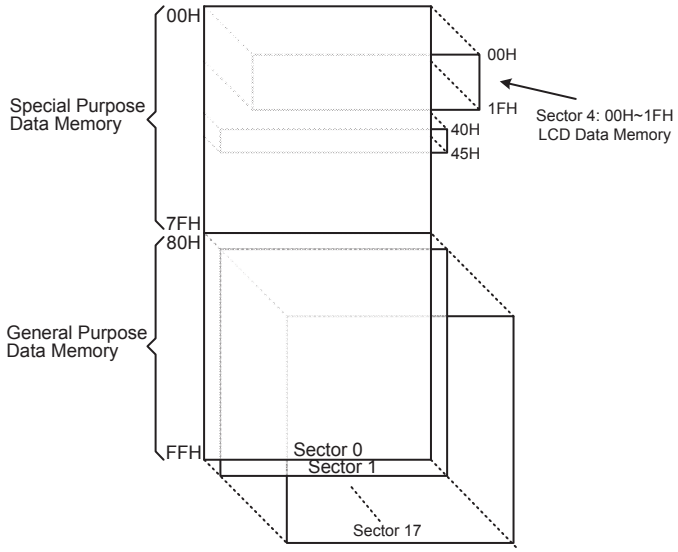
Categorized into two types, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some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

Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide Memory. Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value. The start address of the Data Memory for all devices 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	2304×8	0: 80H~FFH 1: 80H~FFH : 17: 80H~FFH

Data Memory Summary



Data Memory Structure

Data Memory Addressing

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

Direct Addressing can be used in all sectors using the corresponding instruction which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address "m" in the extended instructions can be 13 valid bits for the 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
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH	PBP	
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	INTC0	
11H	INTC1	
12H	INTC2	
13H		
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	PB	
19H	PBC	
1AH	PBPU	
1BH	PC	
1CH	PCC	
1DH	PCPU	
1EH		
1FH		
20H		
21H	PSC0R	
22H	PSC1R	
23H		
24H		
25H		
26H		
27H		
28H		
29H		
2AH		
2BH		
2CH		
2DH		
2EH	RSTC	
2FH	PMPS	
30H	LVPUC	
31H		
32H		
33H	MFI0	
34H	MFI1	
35H	MFI2	
36H		
37H		
38H		
39H	INTEG	
3AH	SCC	
3BH	HIRCC	
3CH		
3DH	LXTC	
3EH	WDTC	
3FH	LVRC	

	Sector 0	Sector 1
40H	LVDC	EEC
41H	EEA	
42H		
43H	EED	FC0
44H		FC1
45H		FC2
46H	PAS0	
47H	PAS1	
48H	PBS0	
49H	PBS1	
4AH	PCS0	
4BH		
4CH		
4DH		
4EH	STMC0	
4FH	STMC1	
50H	STMDL	
51H	STMDH	
52H	STMAL	
53H	STMAH	
54H	STMRP	
55H	CTM0C0	
56H	CTM0C1	
57H	CTM0DL	
58H	CTM0DH	
59H	CTM0AL	
5AH	CTM0AH	
5BH		
5CH		
5DH	CTM1C0	
5EH	CTM1C1	
5FH	CTM1DL	
60H	CTM1DH	
61H	CTM1AL	
62H	CTM1AH	
63H		
64H	TB0C	
65H	TB1C	
66H		
67H	SIMC0	
68H	SIMC1/UUCR1	
69H	SIMC2/SIMA/UUCR2	
6AH	SIMD/UTXR_RXR	
6BH	SIMTOC/UBRG	
6CH	UUSR	
6DH	SPIAC0	
6EH	SPIAC1	
6FH	SPIAD	
70H		
71H		
72H		
73H		
74H	FARL	
75H	FARH	
76H	FD0L	
77H	FD0H	
78H	FD1L	
79H	FD1H	
7AH	FD2L	
7BH	FD2H	
7CH	FD3L	
7DH	FD3H	
7EH		
7FH	LCDC	

: Unused, read as 00H

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 mp1h, a
     mov a, offset adres1    ; Accumulator loaded with first RAM address
     mov mp1l, a             ; setup memory pointer with first RAM address
loop:
     clr IAR1                ; clear the data at address defined by MP1L
     inc mp1l                ; increment memory pointer MP1L
     sdz block               ; check if last memory location has been cleared
     jmp loop
continue:
```

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

Direct Addressing Program Example using extended instructions

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

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

Program Memory Bank Pointer – PBP

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

- **PBP Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **PBP0**: Program Memory Bank Point bit
0: Bank 0
1: Bank 1

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

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

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

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

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

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

- **STATUS Register**

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

"x": unknown

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

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

For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.

For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation zero flag.

For other instructions, the CZ flag will not be affected.

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

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

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

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

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

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

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

EEPROM Data Memory

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 was as any other Special Function Register. The EEC register however, being located in Sector 1, can be read from or written to indirectly using the 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: 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.

Reading Data from the EEPROM

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

Writing Data to the EEPROM

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

Write Protection

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

EEPROM Interrupt

The EEPROM write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. However as the EEPROM is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM write cycle ends, the DEF request flag and its associated multi-function interrupt request flag will both be set. If the global, EEPROM and Multi-function interrupts are enabled and the stack is not full, a jump to the associated Multi-function Interrupt vector will take place. When the interrupt is serviced only the Multi-function interrupt flag will be automatically reset, the EEPROM interrupt flag must be manually reset by the application program. 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/write
CLR  MP1H
MOV  A, EED               ; move read data to register
MOV  READ_DATA, A
```

Writing Data to the EEPROM – polling method

```
MOV  A, EEPROM_ADRES      ; user defined address
MOV  EEA, A
MOV  A, EEPROM_DATA       ; user defined data
MOV  EED, A
MOV  A, 040H              ; setup memory pointer 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  IAR1                 ; disable EEPROM read/write
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, Time Base Interrupts as well as LCD Driver. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options are selected through 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.

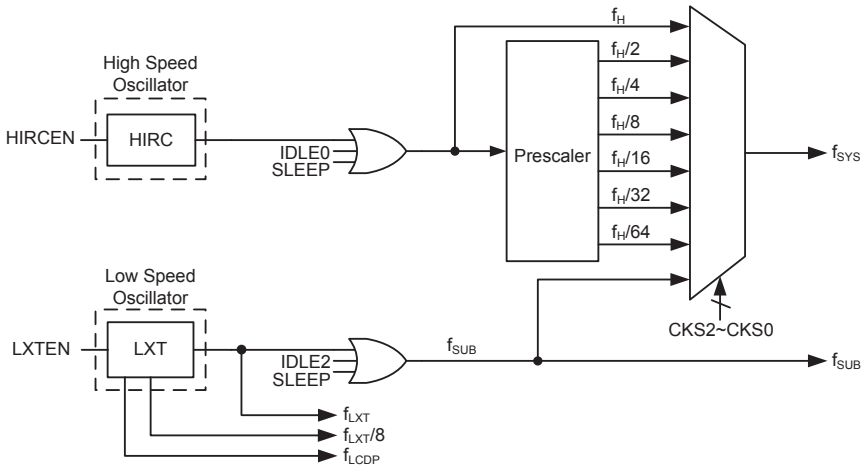
Type	Name	Freq.	Pins
Internal High Speed RC	HIRC	4/8/12MHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

System Clock Configurations

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

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



System Clock Configurations

Internal RC Oscillator – HIRC

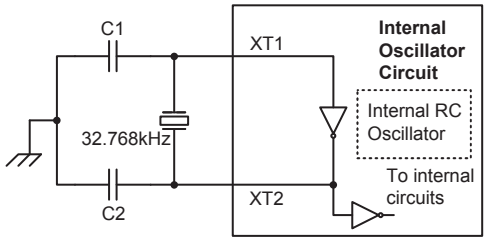
The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 4/8/12MHz, which is selected using a configuration option. The HIRC1~HIRC0 bits in the HIRCC register must also be setup to match the selected configuration option frequency. Setting up these bits is necessary to ensure that the HIRC frequency accuracy specified in the A.C. Characteristics is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 3V or 5V and at a temperature of 25°C degrees, the selected trimmed oscillation frequency will have a tolerance within 1%.

External 32.768kHz Crystal Oscillator – LXT

The External 32.768kHz Crystal System Oscillator is the low frequency oscillator choice, which is always enabled. This clock source has a frequency of 32.768kHz and requires a 32.768kHz crystal to be connected between pins XT1 and XT2. In addition of supplying f_{LXT} with the frequency of 32.768kHz, the clock source provides a divided version of $f_{LXT}/8$ for the Watchdog Timer, Time Base function as well as LCD Driver function. It should be noted that the LXT oscillator also generates an f_{LCDP} clock for the LCD pump, the frequency is determined by the LCDPCK2~LCDPCK0 bits in the LCDC register, refer to the "LCD Register" section for more information. The external capacitor components connected to the 32.768kHz crystal are necessary to provide oscillation. For applications where precise frequencies are essential, these components may be required to provide frequency compensation due to different crystal manufacturing tolerances. After the LXT oscillator is enabled, there is a time delay associated with the LXT oscillator waiting for it to start-up.

However, for some crystals, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.

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



Note: 1. C1 and C2 are required.
2. Although not shown pins have a parasitic capacitance of around 7pF.
3. Although not shown the oscillator circuit has several output frequencies of f_{LXT}, $f_{LXT}/8$ and f_{LCDP}, in which the f_{LCDP} frequency is determined by the LCDPCK[2:0] bits.

External LXT Oscillator

LXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
32.768kHz	7pF	7pF
Note: C1 and C2 values are for guidance only.		

32.768kHz Crystal Recommended Capacitor Values

Internal 32kHz Oscillator – LIRC

The Internal 32kHz Oscillator is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Note that the internal 32kHz oscillator is only used as clock source for power on reset, low voltage detector and low voltage reset functions. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 5V and at a temperature of 25˚C degrees, the fixed oscillation frequency of 32kHz will have a tolerance within 5%.

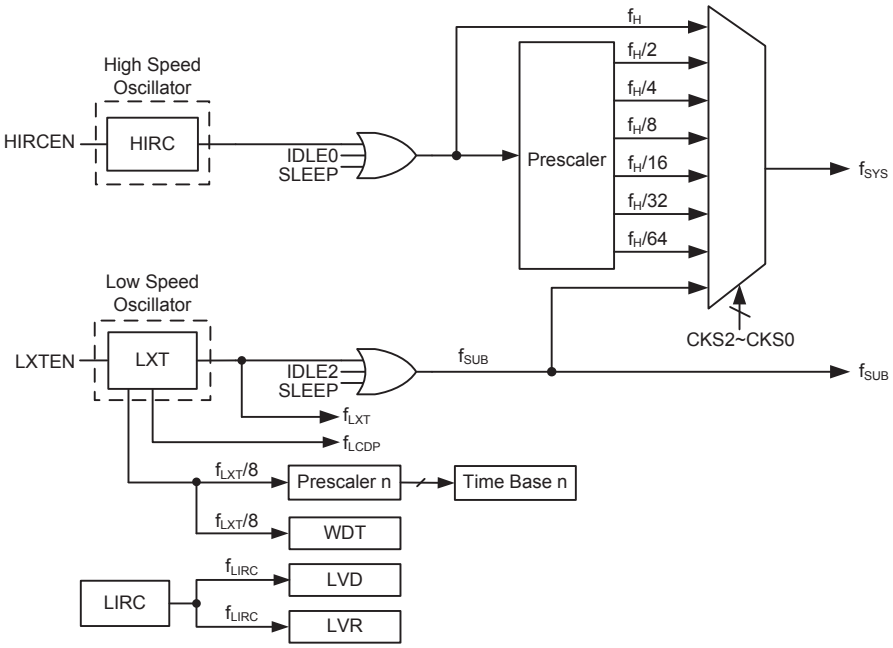
Operating Modes and System Clocks

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 the 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 LXT oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$. Note that the LIRC oscillator is only used for power on reset, low voltage detector and low voltage reset functions. The LXT oscillator is always on even when the device enters the SLEEP mode and the WDT function is turned off.



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~$f_H/64$, 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_{LXT}
		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

"x": don't care

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

FAST Mode

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

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from the LXT 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, and the f_{SUB} clock to peripheral will be stopped too. However if the WDT function is enabled or disabled is determined by the WDTC register.

IDLE0 Mode

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

IDLE1 Mode

The IDLE1 Mode is entered when 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, HIRCC and LXTC, are used to control the system clock and the corresponding oscillator configurations.

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

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC Frequency selection
00: 4MHz
01: 8MHz
10: 12MHz
11: 4MHz
When the HIRC oscillator is enabled or the HIRC frequency selection is changed by application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1.
It is recommended that the HIRC frequency selected by these two bits should be the same with the frequency determined by the configuration options to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator or the HIRC frequency selection is changed by application program, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

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

- **LXTC Register**

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

Bit 7~2 Unimplemented, read as "0"

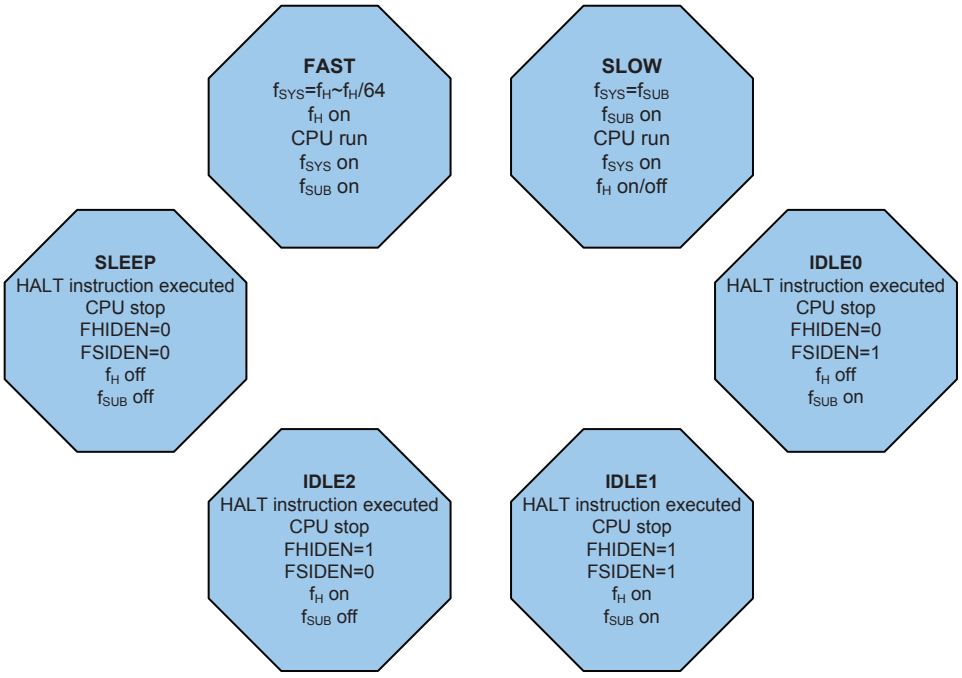
Bit 1 **LXTF**: LXT oscillator stable flag
0: LXT unstable
1: LXT stable
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXT oscillator is enabled, the LXTF bit will first be cleared to 0 and then set to 1 after the LXT oscillator is stable.

Bit 0 **LXTEN**: LXT oscillator enable control (Read only)
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.

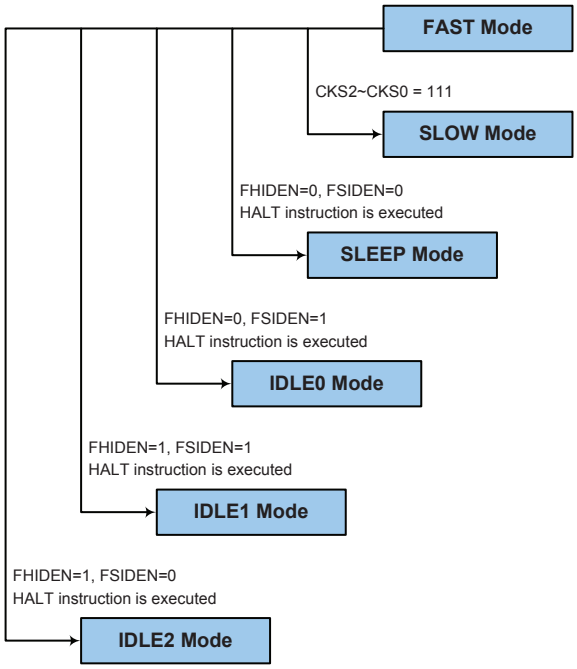
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When an HALT instruction is executed, whether the device enters the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

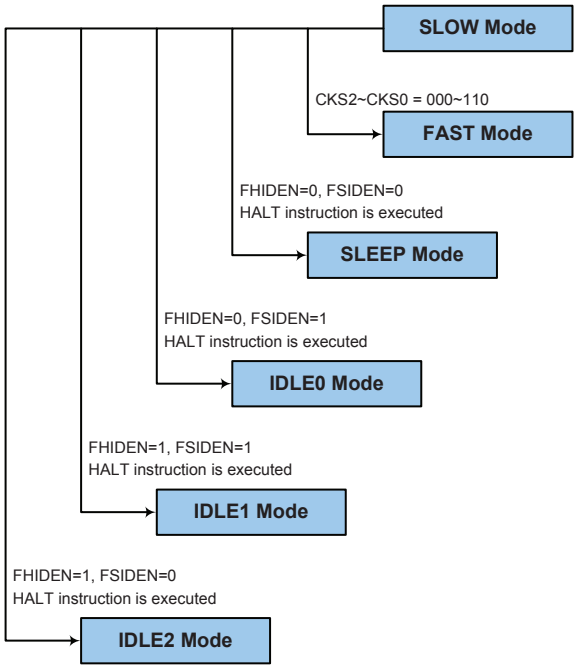
The SLOW Mode is sourced from the LXT oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

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

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



Entering the SLEEP Mode

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

- The system clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled by the WDTC register. 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 if the WDT function is enabled by the WDTC register. 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 if the WDT function is enabled by the WDTC register. 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 if the WDT function is enabled by the WDTC register. 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 unbonbed pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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

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

Wake-up

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

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

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

If the system is woken up by an external reset, the device will experience a full system reset. 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_{WDT}, which is supplied by the LXT oscillator with the output frequency of $f_{LXT}/8$. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable 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	0

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: $2^8/f_{WDT}$
001: $2^{10}/f_{WDT}$
010: $2^{12}/f_{WDT}$
011: $2^{14}/f_{WDT}$
100: $2^{15}/f_{WDT}$
101: $2^{16}/f_{WDT}$
110: $2^{17}/f_{WDT}$
111: $2^{18}/f_{WDT}$
These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period. The internal clock, f_{WDT} is supplied by the LXT oscillator with the output frequency of $f_{LXT}/8$.

- **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
Described elsewhere

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

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

Bit 0 **WRF**: WDT Control Register Software Reset Flag
0: Not 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 instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control and reset control of the Watchdog Timer. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B while the WDT function will be enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after a delay time, t_{SRESET}. After power on these bits will have a value of 01010B.

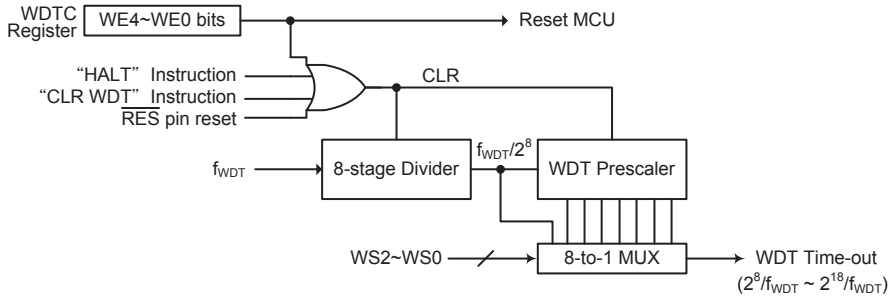
WE4 ~ WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other values	Reset MCU

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. Four 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. The last is an external hardware reset, which means a low level on the external reset pin if the external reset pin exists by the RSTC register.

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 ration.



Watchdog Timer

Reset and Initialisation

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

In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is already running, the $\overline{RES}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to preceed with normal operation after the reset line is allowed to return high.

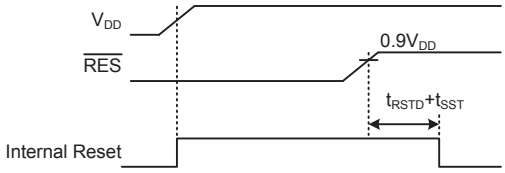
The Watchdog Timer overflow is one of many reset types and will reset the microcontroller. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{RES}$ reset is implemented in situations where the power supply voltage falls below a certain threshold.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

Reset Functions

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

Power-on Reset

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

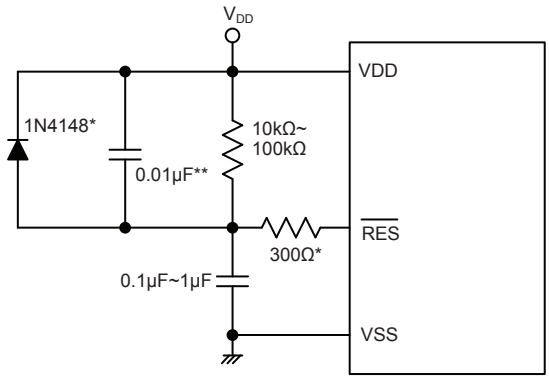


Power-On Reset Timing Chart

$\overline{RES}$ Pin Reset

As the reset pin is shared with I/O pins, the reset function must be selected using a control register, RSTC. Although the microcontroller has an internal RC reset function, if the V_{DD} power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{RES}$ pin, whose additional time delay will ensure that the $\overline{RES}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{RES}$ line reaches a certain voltage value, the reset delay time, t_{RSTD}, is invoked to provide an extea delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Time. For most applications a resistor connected between VDD and the $\overline{RES}$ line and a

capacitor connected betweeb VSS and the $\overline{RES}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{RES}$ pin should be kept as short as possible to minimise any stray noise interference. For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.

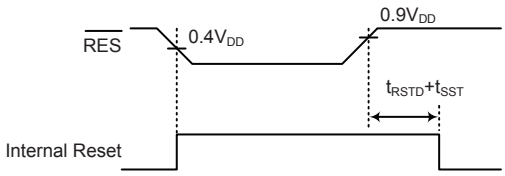


Note: "*" It is recommended that this component is added for added ESD protection.

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

External $\overline{RES}$ Circuit

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



$\overline{RES}$ Reset Timing Chart

There is an internal reset control register, RSTC, which is used to select the external $\overline{RES}$ pin function and 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	PB0
10101010B	$\overline{RES}$
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: PB0
10101010: $\overline{\text{RES}}$ pin
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET}, and the RSTF bit in the RSTFC register will be set to 1.
All resets will reset this register to POR value except the WDT time out hardware warm reset. Note that if the register is set to 10101010 to select the $\overline{\text{RES}}$ pin, this configuration has higher priority than other related pin-shared controls.

- **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.

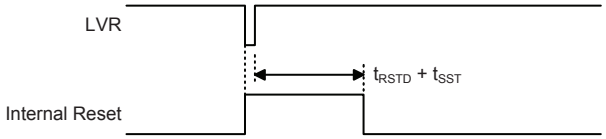
Bit 2 **LVRF**: LVR function reset flag
Described elsewhere

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

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

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the 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~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 01100110B. Note that the LVR function will be automatically disabled when the device enters the IDLE/SLEEP mode.



Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select
01100110B: 1.7V
01010101B: 1.9V
00110011B: 2.55V
10011001B: 3.15V
10101010B: 3.8V
11110000B: LVR disable
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
Described elsewhere

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

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

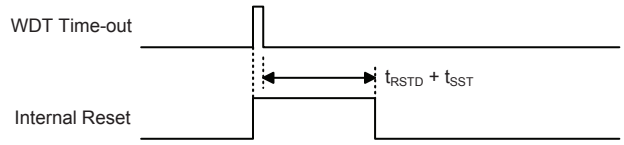
Bit 0 **WRF**: WDT Control register software reset flag
Described elsewhere

In Application Programming Reset

The device contains an IAP function, therefore an IAP reset exists, which is caused by writing data 55H to FC1 register.

Watchdog Time-out Reset during Normal Operation

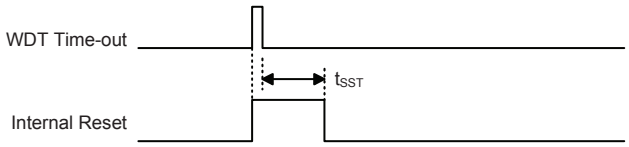
The Watchdog time-out Reset during normal operation is the same as LVR reset except that the Watchdog time-out flag TO will be set high.



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	$\overline{RES}$ or LVR reset during FAST or SLOW Mode operation
1	u	WDT time-out reset during FAST or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

"u" stands for unchanged

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

Item	Condition after Reset
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Bases	Clear after reset, WDT begins counting
Timer 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.

Register	Reset (Power On)	$\overline{\text{RES}}$ Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	xx00 xxxx	uuuu uuuu	uuuu uuuu	uu1u uuuu	uu11 uuuu
PBP	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
IAR2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- 0x00	---- uuuu	---- u1uu	---- uuuu	---- uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PCC	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PCPU	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
PSC0R	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
PSC1R	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
RSTC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
PMPS	---- --00	---- --00	---- --00	---- --00	---- --uu
LVPUC	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
MFI0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI1	--00 --00	--00 --00	--00 --00	--00 --00	--uu --uu
MFI2	--00 --00	--00 --00	--00 --00	--00 --00	--uu --uu
INTEG	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SCC	001- --00	001- --00	001- --00	001- --00	uuu- --uu
HIRCC	---- 0001	---- 0001	---- 0001	---- 0001	---- uuuu
LXTC	---- --01	---- --01	---- --01	---- --01	---- --uu
WDTC	0101 0010	0101 0010	0101 0010	0101 0010	uuuu uuuu
LVRC	0110 0110	0110 0110	0110 0110	0110 0110	uuuu uuuu

Register	Reset (Power On)	$\overline{\text{RES}}$ Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
LVDC	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
EEA	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
EED	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS0	00-- 00--	00-- 00--	00-- 00--	00-- 00--	uu-- uu--
PAS1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	00-- 00--	00-- 00--	00-- 00--	00-- 00--	uu-- uu--
PBS1	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
PCS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMC0	0000 0---	0000 0---	0000 0---	0000 0---	uuuu u---
STMC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDH	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAH	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMRP	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0C0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0C1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DH	---- --00	---- --00	---- --00	---- --00	---- --uu
CTM0AL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0AH	---- --00	---- --00	---- --00	---- --00	---- --uu
CTM1C0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1C1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DH	---- --00	---- --00	---- --00	---- --00	---- --uu
CTM1AL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1AH	---- --00	---- --00	---- --00	---- --00	---- --uu
TB0C	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu
SIMC0	1110 0000	1110 0000	1110 0000	1110 0000	uuuu uuuu
SIMC1 (UMD=0)	1000 0001	1000 0001	1000 0001	1000 0001	uuuu uuuu
UUCR1* (UMD=1)	0000 00x0	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
SIMD/ UTXR_RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2/ UUCR2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMTOC (UMD=0)	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
UBRG* (UMD=1)	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
UUSR	0000 1011	0000 1011	0000 1011	0000 1011	uuuu uuuu
SPIAC0	111- --00	111- --00	111- --00	111- --00	uuu- --uu
SPIAC1	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
SPIAD	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
FARL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
FD0L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	Reset (Power On)	$\overline{RES}$ Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
FD0H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
LCDC	0-00 0000	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
EEC	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
FC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC2	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u

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

Input/Output Ports

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	PAC5	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU4	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	—	—	—	—	PC3	PC2	PC1	PC0
PCC	—	—	—	—	PCC3	PCC2	PCC1	PCC0
PCPU	—	—	—	—	PCPU3	PCPU2	PCPU1	PCPU0
LVPUC	—	—	—	—	—	—	—	LVPU

"—": Unimplemented

I/O Logic Function Register List

Pull-high Resistors

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

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

- **PxPU Register**

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

PxPUn: I/O Port x Pin Pull-high Function Control
0: Disable
1: Enable

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

- **LVPUC Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **LVPU**: Pull-high resistor select when low voltage power supply
0: All pin pull high resistor is 60kΩ (typ.) @ 3V
1: All pin pull high resistor is 15kΩ (typ.) @ 3V

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

Port A Wake-up

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

Note that the wake-up function can be controlled by the wake-up control registers only when the pin-shared functional pin is selected as general purpose input/output 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

PAWUn: Port A Pin Wake-up Control
0: Disable
1: Enable

I/O Port Control Registers

Each I/O port has its own control register which controls 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	PxC5	PxC5	PxC4	PxC3	PxC2	PxC1	PxC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

PxCn: I/O Port x Pin Type Selection
0: Output
1: Input
The PxCn bit is used to control the pin type selection. Here the "x" can be A, B and C. However, the actual available bits for each I/O port may be different.

I/O Port Power Source Control

The device supports different I/O port power source selections for PC3~PC0. The port power can come from either the power pin VDD or VDDIO which is determined using the PMPS1~PMPS0 bits in the PMPS register. The VDDIO power pin function should first be selected using the corresponding pin-shared function selection bits if the port power is supposed to come from the VDDIO pin. An important point to know is that the input power voltage on the VDDIO pin should be equal to or less than the device supply power voltage when the VDDIO pin is selected as the port power supply pin.

- **PMPS Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PMPS1~PMPS0**: PC3~PC0 pin power source selection
0x: VDD
1x: VDDIO
If the PA3 pin-shared function is switched to the VDDIO function, the VDDIO input voltage can be used as the PC3~PC0 I/O port power source by setting the PMPS[1:0] bit field to "1x". Note that the input power voltage on the VDDIO pin should be equal to or less than the device supply power voltage.

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, special point must be noted for some digital input pins, such as INTn, xTCKn, 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	—	—	PBS03	PBS02	—	—
PBS1	—	—	—	—	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00

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
00/01: PA3
10: STP
11: VDDIO

Bit 5~4 Unimplemented, read as "0"

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00/01/10: PA1
11: CTP1

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
00/01/10: PA7
11: SCKA

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

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

Bit 1~0 **PAS11~PAS10**: PA4 pin-shared function selection
00/01/10: PA4
11: $\overline{\text{SCSA}}$

- **PBS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	—	—	PBS03	PBS02	—	—
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
00/01/10: PB3
11: CTP0

Bit 5~4 Unimplemented, read as "0"

Bit 3~2 **PBS03~PBS02**: PB1 pin-shared function selection
00/01/10: PB1
11: STPB

Bit 1~0 Unimplemented, read as "0"

- **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PBS13	PBS12	PBS11	PBS10
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 **PBS13~PBS12**: PB5 pin-shared function selection
00/01/10: PB5/INT1
11: CTP1B

Bit 1~0 **PBS11~PBS10**: PB4 pin-shared function selection
00/01/10: PB4/INT0
11: CTP0B

- **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PCS07~PCS06**: PC3 pin-shared function selection
00/01/10: PC3
11: SCK/SCL

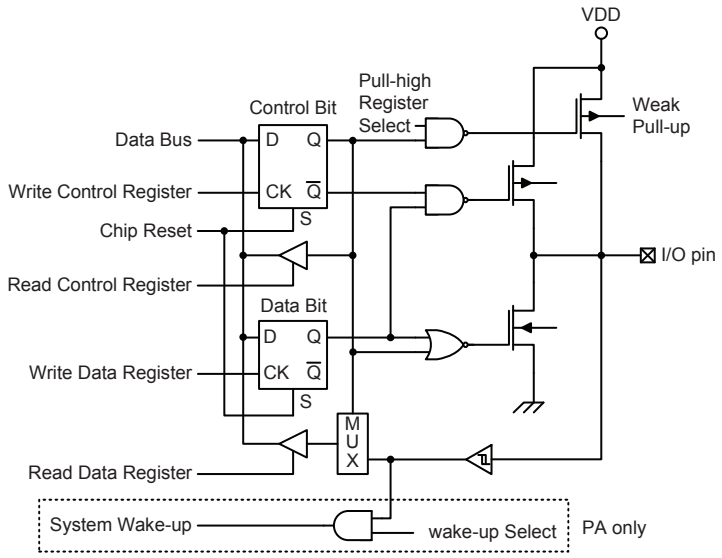
Bit 5~4 **PCS05~PCS04**: PC2 pin-shared function selection
00/01/10: PC2
11: SDO/TX

Bit 3~2 **PCS03~PCS02**: PC1 pin-shared function selection
00/01/10: PC1
11: SDI/SDA/RX

Bit 1~0 **PCS01~PCS00**: PC0 pin-shared function selection
00/01/10: PC0
11: $\overline{SCS}$

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.



Logic Function Input/Output Structure

Programming Considerations

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

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

Timer Modules – TM

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

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

Introduction

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

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

TM Function Summary

TM Operation

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

TM Clock Source

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

TM Interrupts

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

TM External Pins

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

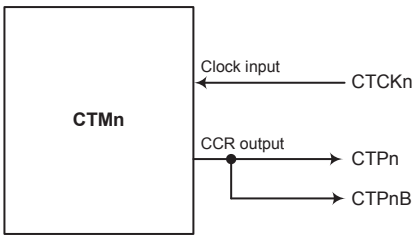
The Standard type TM has another input pin, STPI, which is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the STIO1~STIO0 bits in the STMC1 register.

The TMs each has two output pins, xTPn and xTPnB. The TM output pin can be selected using the corresponding pin-shared function selection bits described in the Pin-shared Function section. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTPn and xTPnB output pin are also the pins where the TM generates the PWM output waveform.

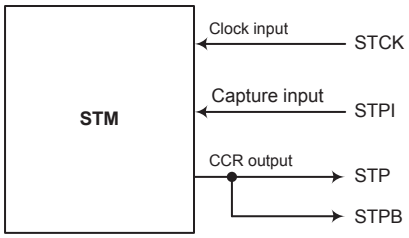
As the TM input/output pins are pin-shared with other functions, the TM input/output function must first be setup using relevant pin-shared function selection register. The details of the pin-shared function selection are described in the pin-shared function section.

CTM		STM	
Input	Output	Input	Output
CTCK0 CTCK1	CTP0, CTP0B CTP1, CTP1B	STCK, STPI	STP, STPB

TM External Pins



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

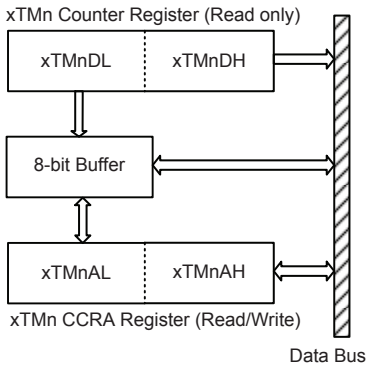


STM Function Pin Block Diagram

Programming Considerations

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

As the CCRA 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 xTMnAL, using the following access procedures. Accessing the CCRA low byte registers without following these access procedures will result in unpredictable values.

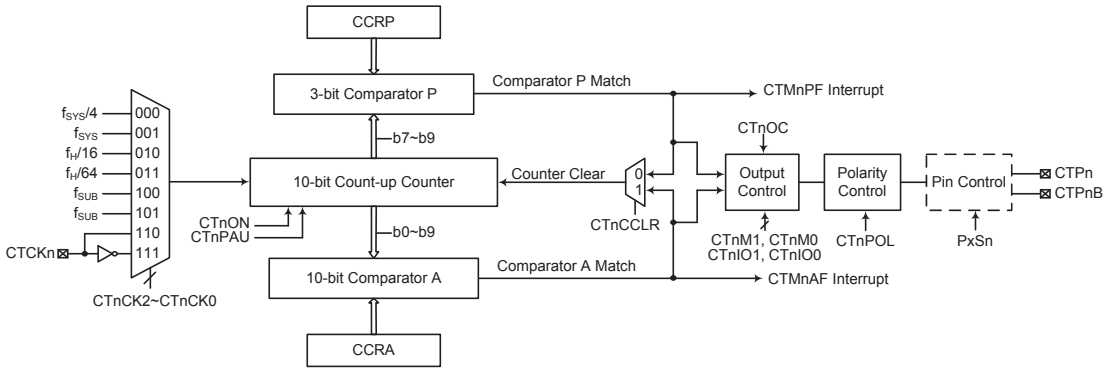


The following steps show the read and write procedures:

- Writing Data to CCRA
  - Step 1. Write data to Low Byte xTMnAL
    – note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte xTMnAH
    – 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 xTMnDH, xTMnAH
    – here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - Step 2. Read data from the Low Byte xTMnDL, xTMnAL
    – this step reads data from the 8-bit buffer.

Compact Type TM – CTM

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



Note: The CTPnB is the inverted output of the CTPn.

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

Compact TM Operation

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

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the 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 CTM interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control one output pin. 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 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 CTPn output 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 Comparatror P match
1: CTMn Comparatror A match

This bit is used to select the method which clears the counter. Remember that the Compact TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the 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

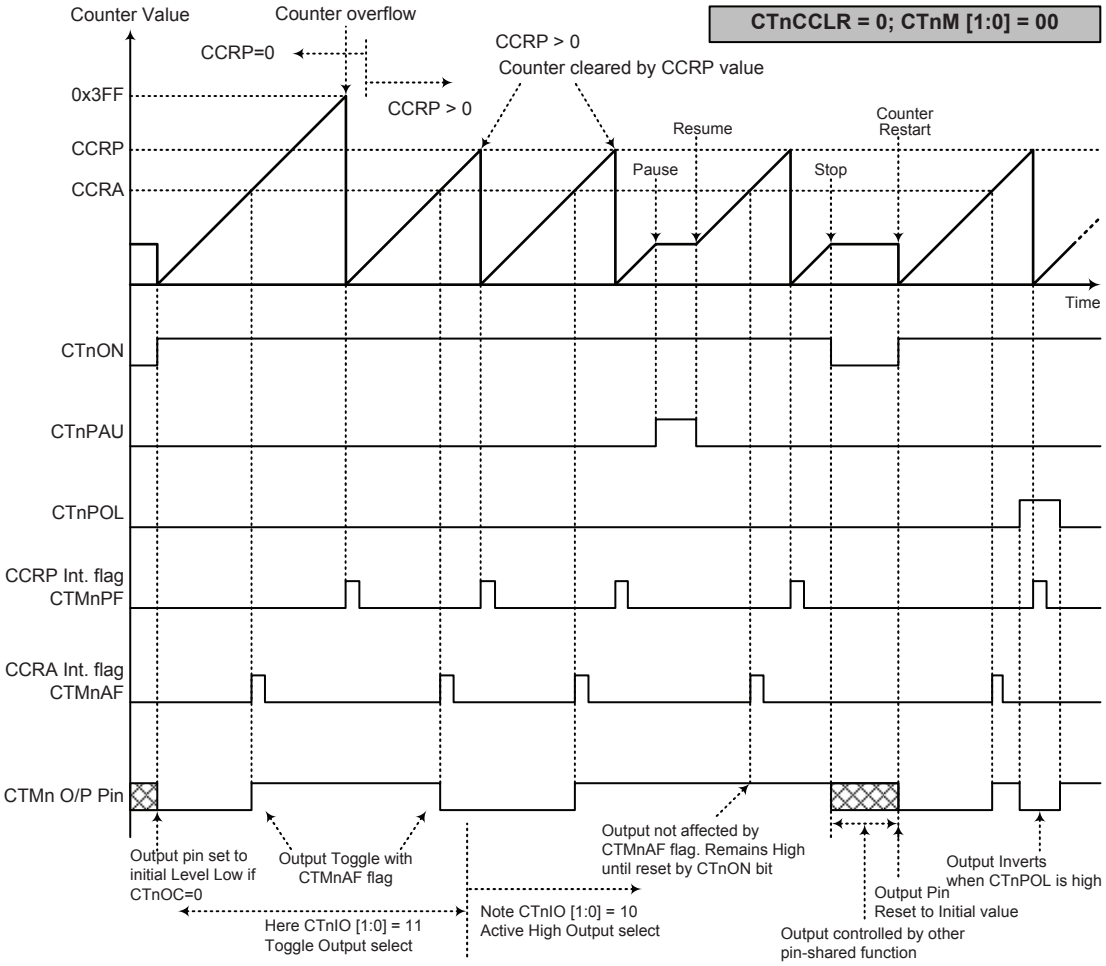
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

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.

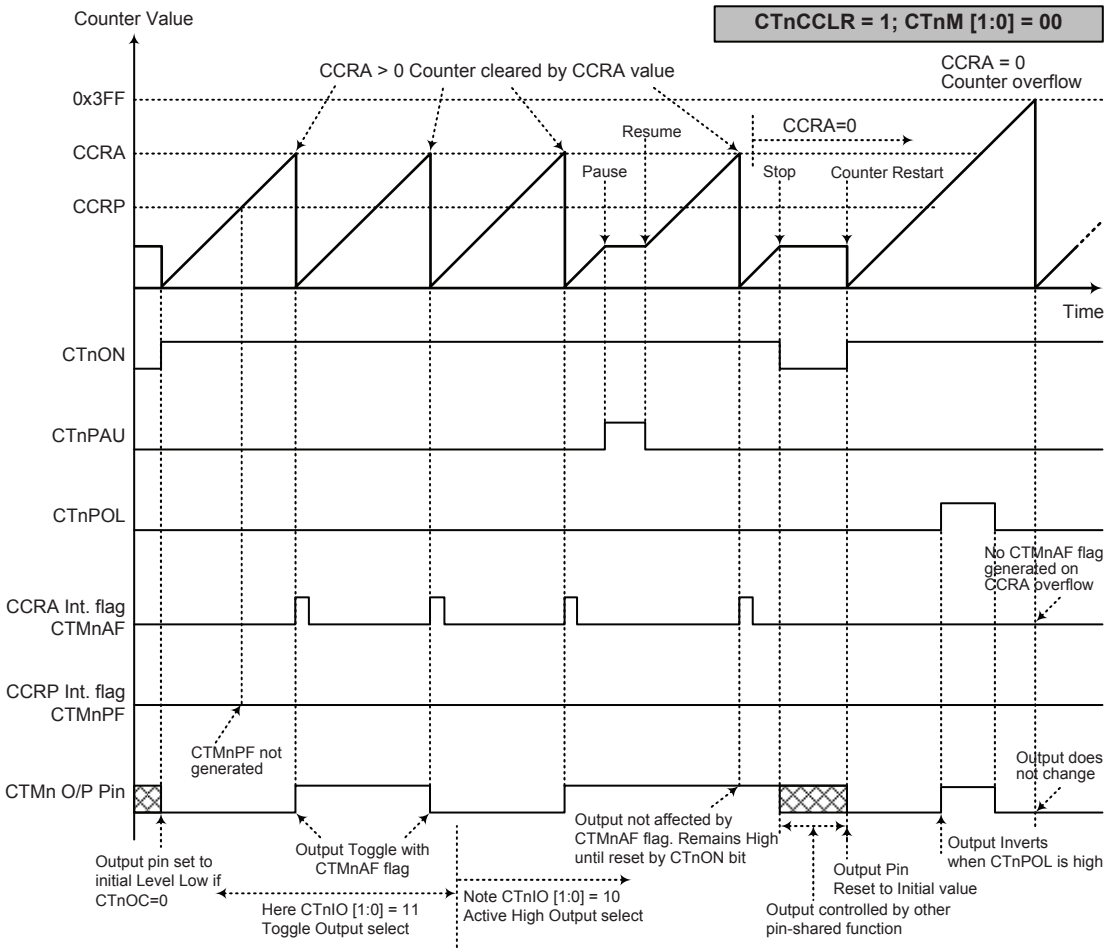
If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is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

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



Compare Match Output Mode – CTnCCLR=0

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



Compare Match Output Mode – CTnCCLR=1

Note: 1. With CTnCCLR=1, a Comparator A match will clear the counter
2. The CTMn output pin 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 CTnCCLR=1
5. n=0~1

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

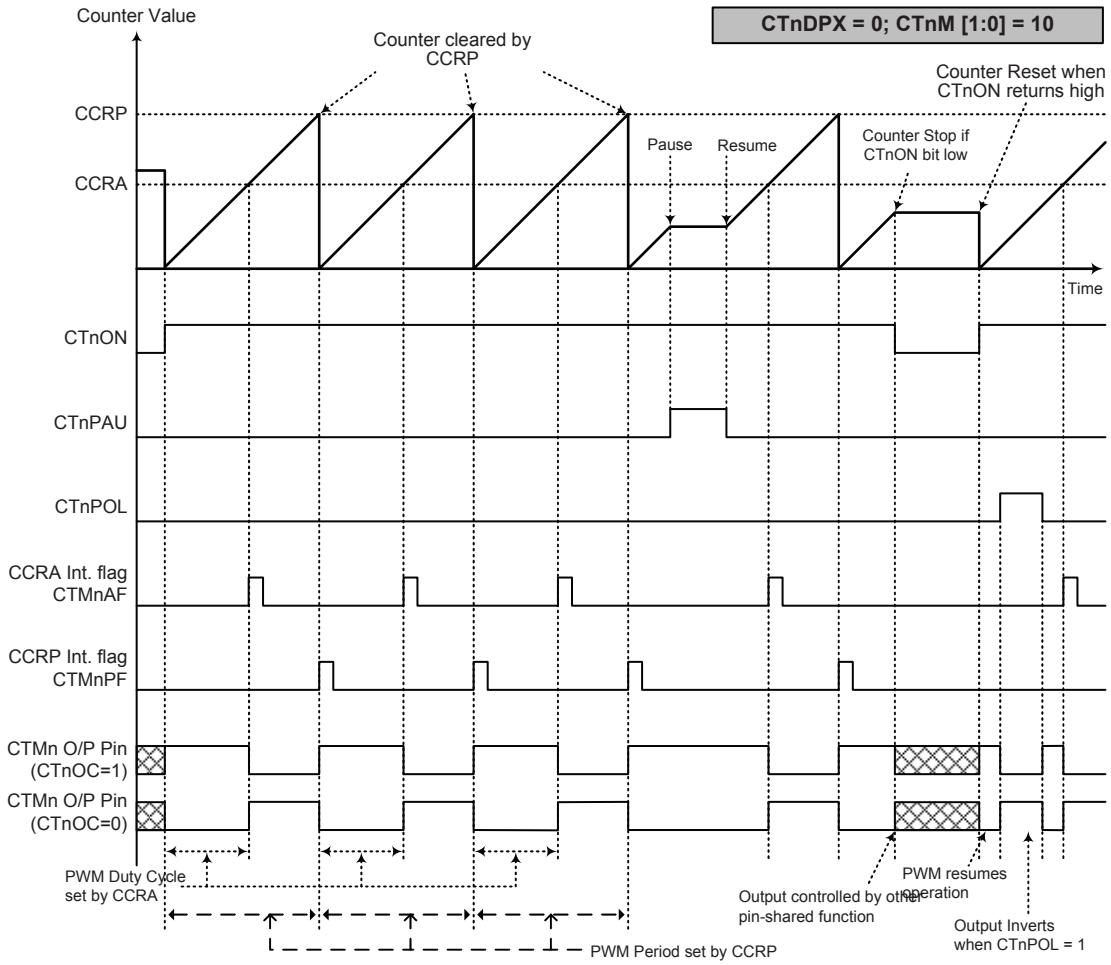
The CTMn PWM output frequency=$(f_{SYS}/4)/(2\times128)=f_{SYS}/1024$=7.812kHz, duty=128/(2×128)=50%.

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

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

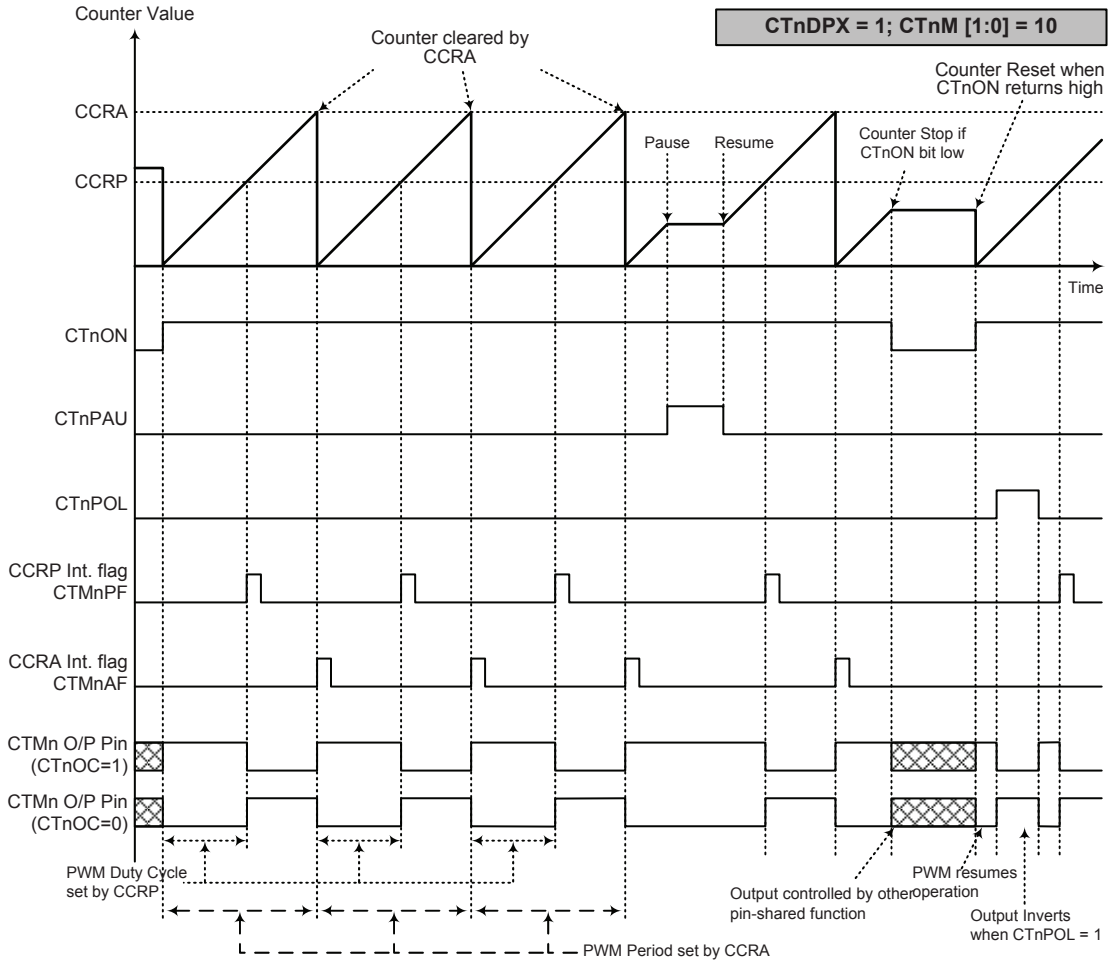
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

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.



PWM Output Mode – CTnDPX=0

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
5. n=0~1

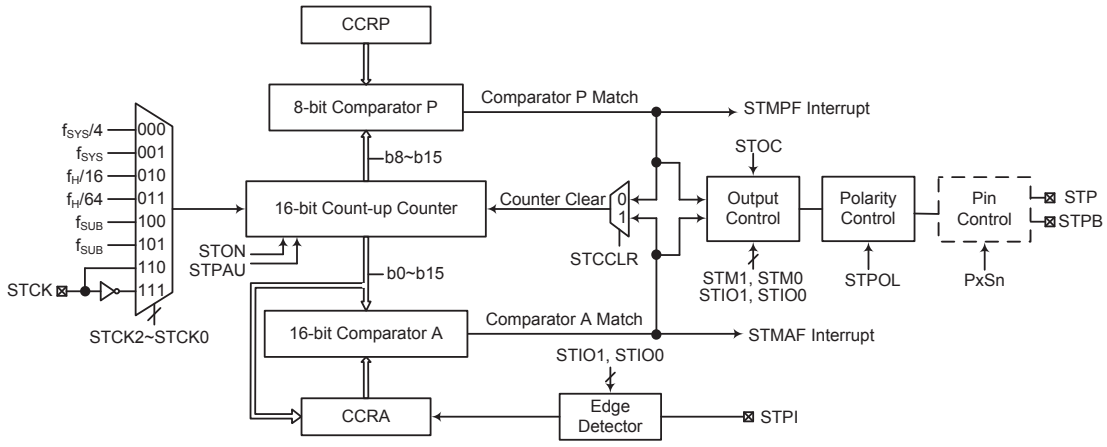


PWM Output Mode – CTnDPX=1

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

Standard Type TM – STM

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



Note: The STPB is the inverted output of the STP.

Standard Type TM Block Diagram

Standard Type TM Operation

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

The only way of changing the value of the 16-bit counter using the application program, is to clear the counter by changing the STON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a STM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control an output pin. All operating setup conditions are selected using relevant internal registers.

Standard Type TM Register Description

Overall operation of the Standard TM is controlled using a series of registers. A read only register pair exists to store the internal counter 16-bit value, while a read/write register pair exists to store the internal 16-bit CCRA value. The STMRP register is used to store the 8-bit CCRP value. The remaining two registers are control registers which setup the different operating and control modes.

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

16-bit Standard Type TM Register List

- **STMDL Register**

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

Bit 7~0 STM Counter Low Byte Register bit 7 ~ bit 0
STM 16-bit Counter bit 7 ~ bit 0

- **STMDH Register**

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

Bit 7~0 STM Counter High Byte Register bit 7 ~ bit 0
STM 16-bit Counter bit 15 ~ bit 8

- **STMAL Register**

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

Bit 7~0 STM CCRA Low Byte Register bit 7 ~ bit 0
STM 16-bit CCRA bit 7 ~ bit 0

- **STMAH Register**

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

Bit 7~0 STM CCRA High Byte Register bit 7 ~ bit 0
STM 16-bit CCRA bit 15 ~ bit 8

- **STMRP Register**

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

Bit 7~0 **D7~D0**: STM CCRP 8-bit register, compared with the STM counter bit 15~bit 8

Comparator P match period =
0: 65536 STM clocks
1~255: (1~255) × 256 STM clocks

These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter›s highest eight bits. The result of this comparison can be selected to clear the internal counter if the STCCLR bit is set to zero. Setting the STCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

- **STMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **STPAU**: STM Counter Pause control
0: Run
1: Pause

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

Bit 6~4 **STCK2~STCK0**: Select STM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: STCK rising edge clock
111: STCK falling edge clock

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

Bit 3 **STON**: STM Counter On/Off control
0: Off
1: On

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

the Compare Match Output Mode, the PWM Output Mode or the Single Pulse Output Mode then the STM output pin will be reset to its initial condition, as specified by the STOC bit, when the STON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

- **STMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **STM1~STM0**: Select STM Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

Bit 5~4 **STIO1~STIO0**: Select STM external pin STP function

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

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

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

Timer/Counter Mode
Unused

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

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

In the PWM Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the STIO1 and STIO0 bits only after the STM has been switched off. Unpredictable PWM outputs will occur if the STIO1 and STIO0 bits are changed when the STM is running.

Bit 3 **STOC**: STM STP Output control

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

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

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

Bit 2 **STPOL**: STM STP Output polarity control
0: Non-inverted
1: Inverted

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

Bit 1 **STDPX**: STM PWM duty/period control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

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

Bit 0 **STCCLR**: STM Counter Clear condition selection
0: Comparator P match
1: Comparator A match

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

Standard Type TM Operation Modes

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

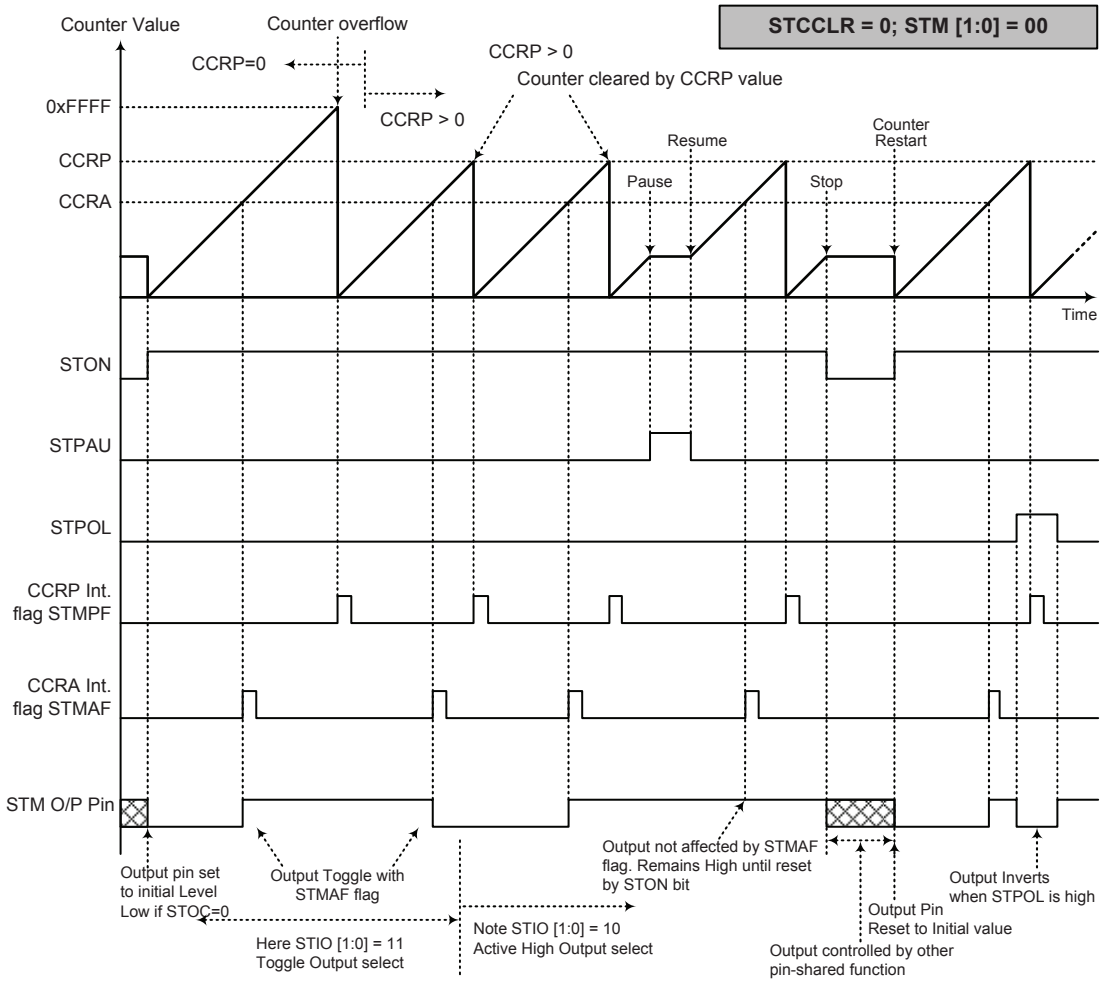
Compare Match Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the STCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both STMAF and STMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

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

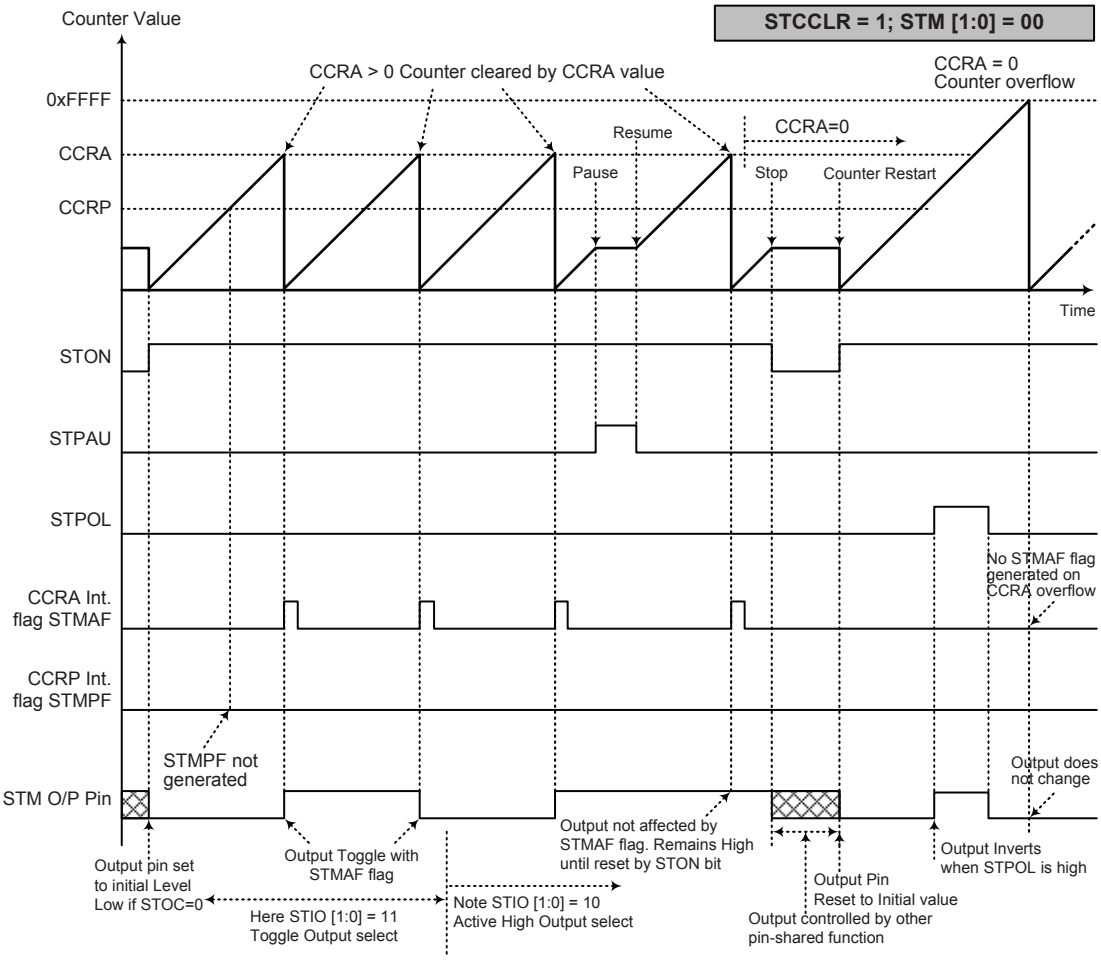
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 16-bit, FFFF Hex, value, however here the STMAF interrupt request flag will not be generated.

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



Compare Match Output Mode – STCCLR=0

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



Compare Match Output Mode – STCCLR=1

Note: 1. With STCCLR=1 a Comparator A match will clear the counter
2. The STM output pin is controlled only by the STMAF flag
3. The output pin is reset to its initial state by a STON bit rising edge
4. A STMPF flag is not generated when STCCLR=1

Timer/Counter Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the STM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the STM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 10 respectively. The PWM function within the STM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the STM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the STCCLR bit has no effect as the PWM period. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the STDPX bit in the STMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The STOC bit in the STMC1 register is used to select the required polarity of the PWM waveform while the two STIO1 and STIO0 bits are used to enable the PWM output or to force the STM output pin to a fixed high or low level. The STPOL bit is used to reverse the polarity of the PWM output waveform.

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

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

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

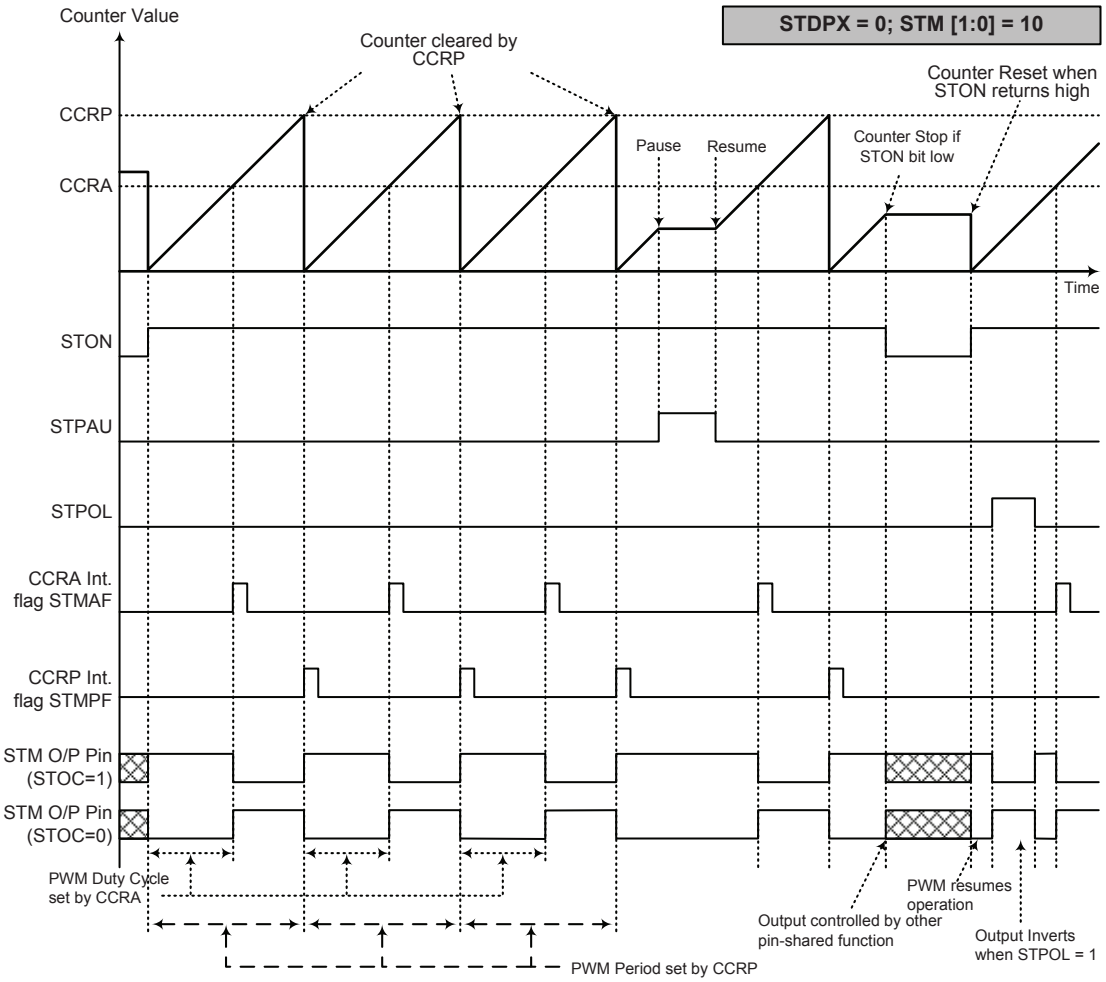
The STM PWM output frequency=(f_{SYS}/4)/(2×256)=f_{SYS}/2048=4kHz, duty=128/(2×256)=25%.

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

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

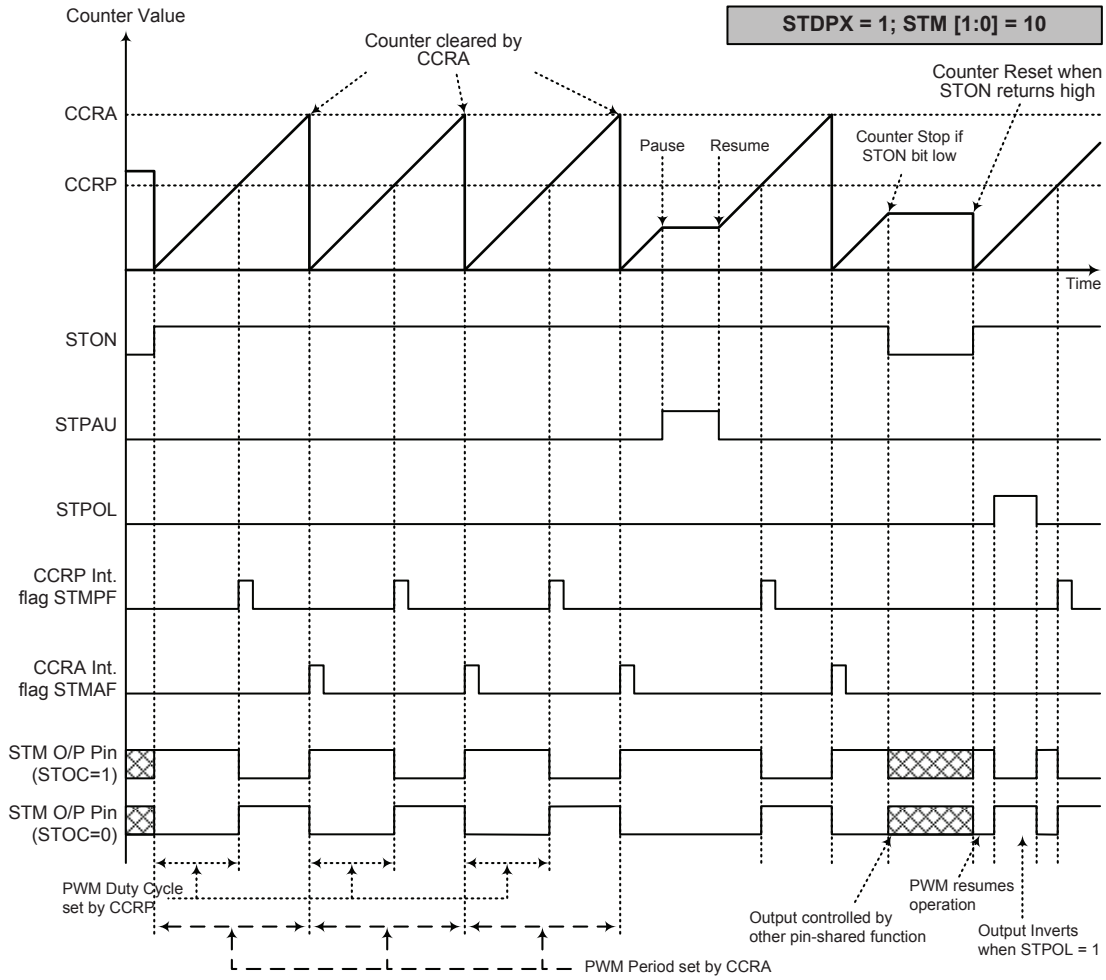
CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

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



PWM Output Mode – STDPX=0

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



PWM Output Mode – STDPX=1

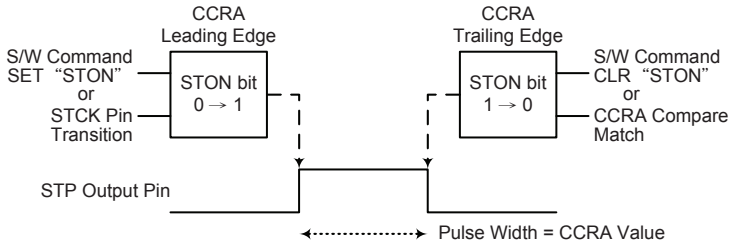
Note: 1. Here STDPX=1 – Counter cleared by CCRA
2. A counter clear sets the PWM Period
3. The internal PWM function continues even when STIO [1:0]=00 or 01
4. The STCCLR bit has no influence on PWM operation

Single Pulse Output Mode

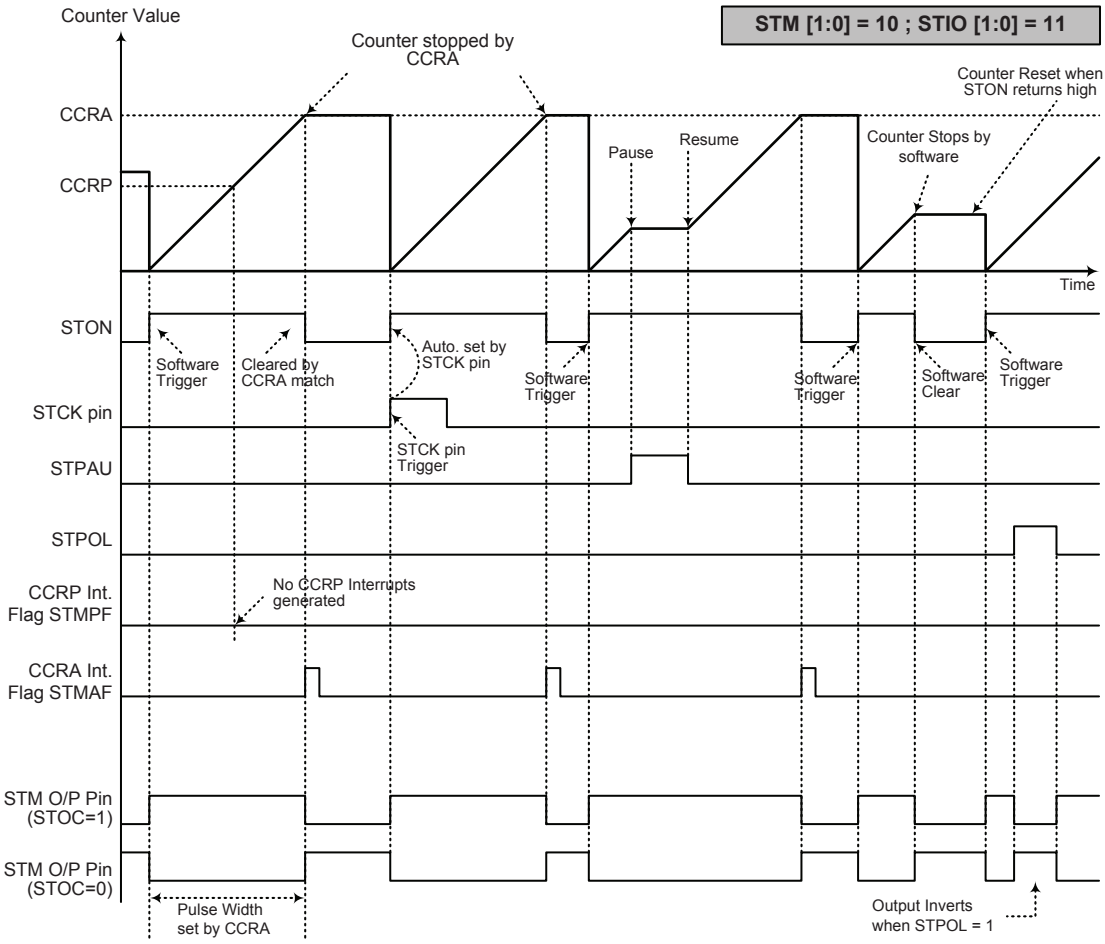
To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

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

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



Single Pulse Generation



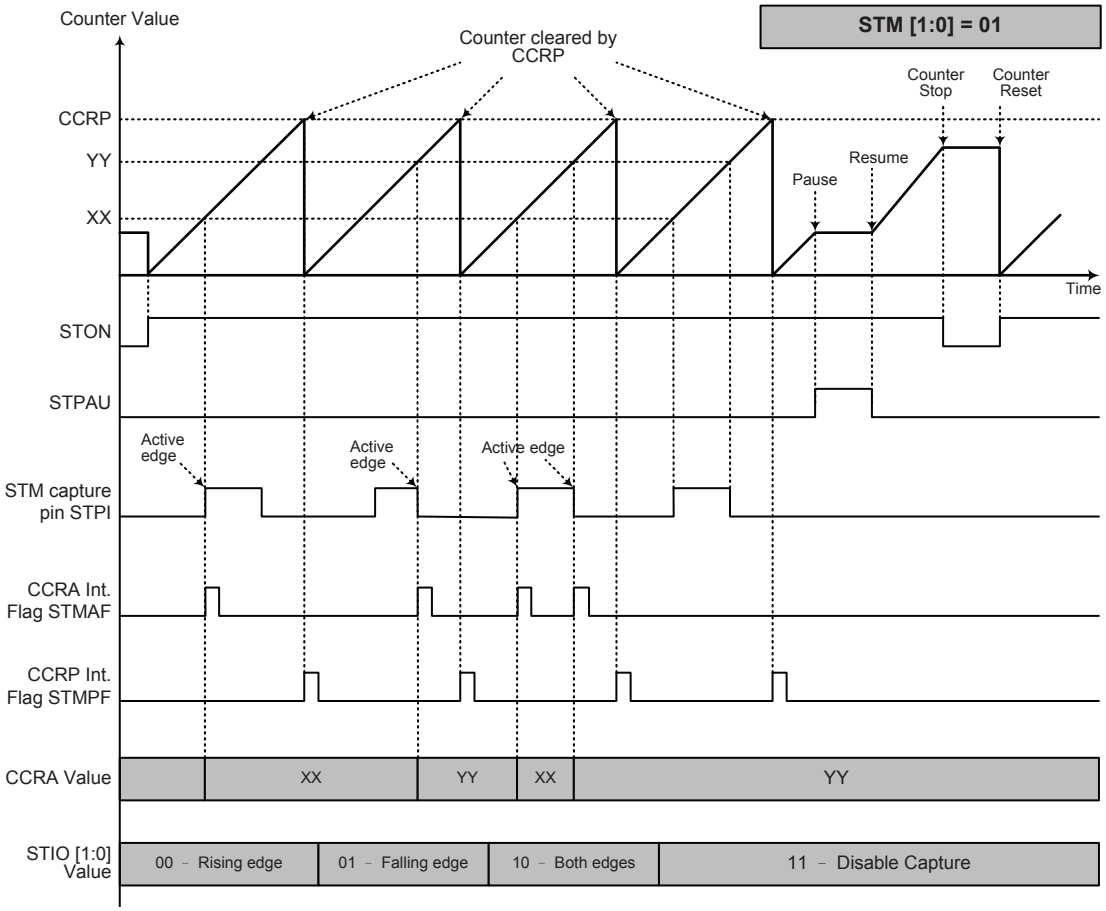
Single Pulse Output Mode

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

Capture Input Mode

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

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



Capture Input Mode

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

Universal Serial Interface Module – USIM

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

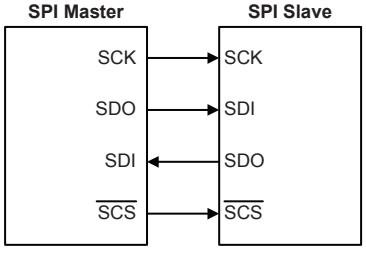
SPI Interface

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

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

SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I²C/UART function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ pin only one slave device can be utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to 1 to enable $\overline{SCS}$ pin function, set CSEN bit to 0 the $\overline{SCS}$ pin will be floating state.

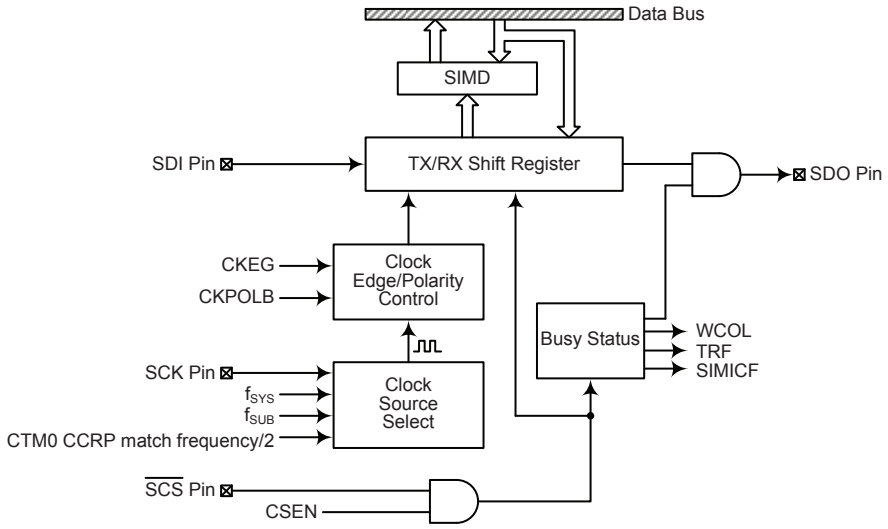


SPI Master/Slave Connection

The SPI function in the device offers the following features:

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

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



SPI Block Diagram

SPI Registers

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

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

SPI Register List

SPI Data Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I^2C functions. Before the device writes data to the SPI bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the SPI bus, the device can read it from the SIMD register. Any transmission or reception of data from the SPI bus must be made via the SIMD register.

- **SIMD Register**

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

"x": unknown

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

SPI Control Registers

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

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is CTM0 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode
When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from CTM0 and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

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

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

Bit 1 **SIMEN**: USIM SPI/I²C Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the USIM SPI/I²C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I²C interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I²C interface is enabled. If the USIM is configured to operate as an SPI interface via the UMD and SIM2~SIM0 bits, the contents of the SPI control registers will remain at the

previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the USIM is configured to operate as an I^2C interface via the UMD and SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I^2C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I^2C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

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

- **SIMC2 Register**

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

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

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

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

Bit 3 **MLS**: SPI data shift order
0: LSB first
1: MSB first
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

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

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

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

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

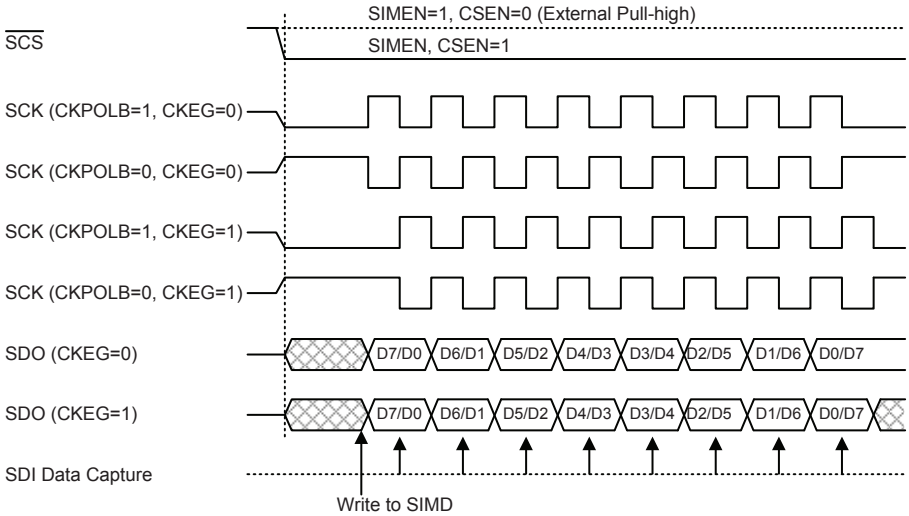
Bit 0 **TRF**: SPI Transmit/Receive complete flag
0: SPI data is being transferred
1: SPI data transfer is completed

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

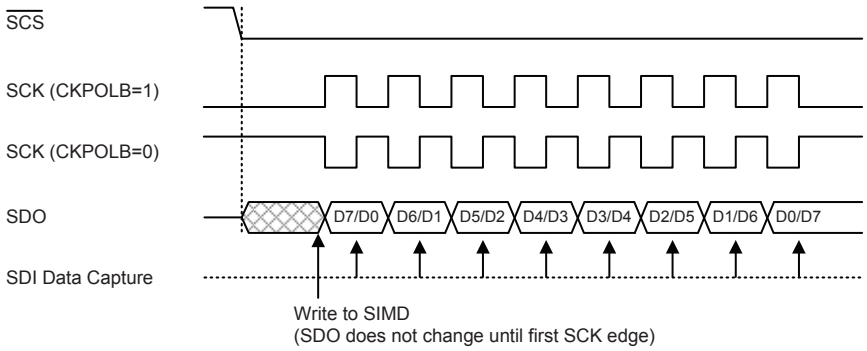
SPI Communication

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

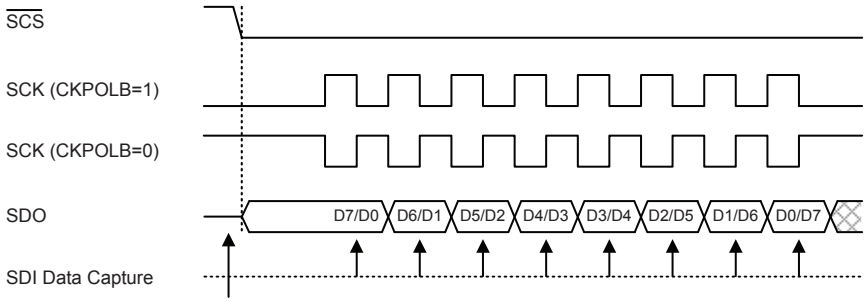
The SPI will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active..



SPI Master Mode Timing

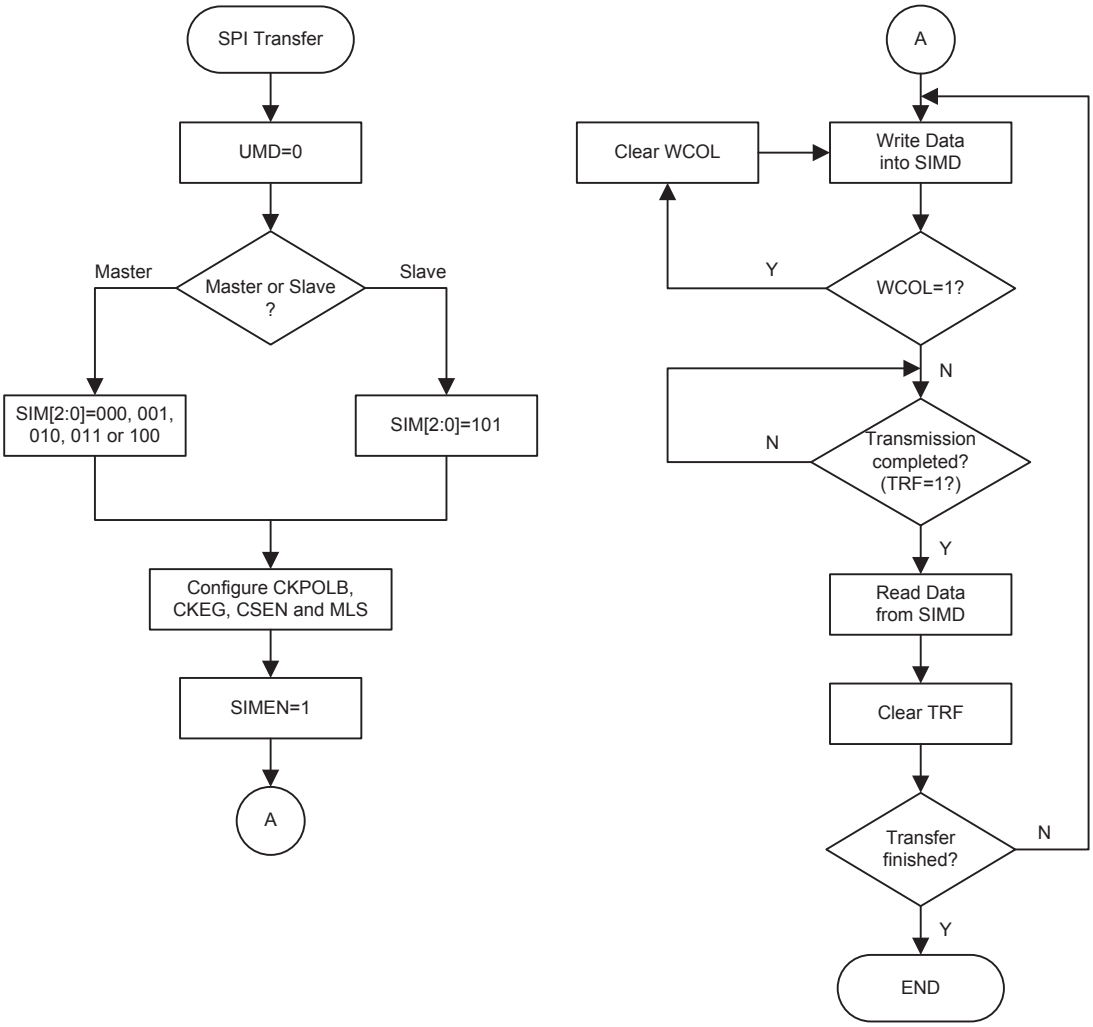


SPI Slave Mode Timing – CKEG=0



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

SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flow Chart

SPI Bus Enable/Disable

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

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

SPI Operation Steps

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

The CSEN bit in the SIMC2 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{SCS}$ line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the $\overline{SCS}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the SDI line in a floating condition and the SDO line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and $\overline{SCS}$, SDI, SDO and SCK will all become I/O pins or the other functions using the corresponding pin-shared control bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SIMD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode

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

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

Slave Mode

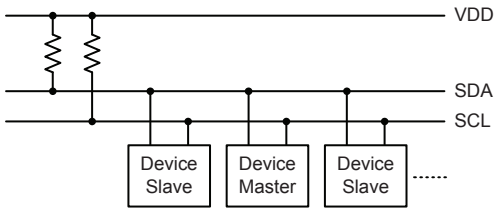
- Step 1
  Select the SPI Slave mode using the UMD and SIM2~SIM0 bits in the SIMC0 control register
- Step 2
  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.
- Step 3
  Setup the SIMEN bit in the SIMC0 control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

Error Detection

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

I²C Interface

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

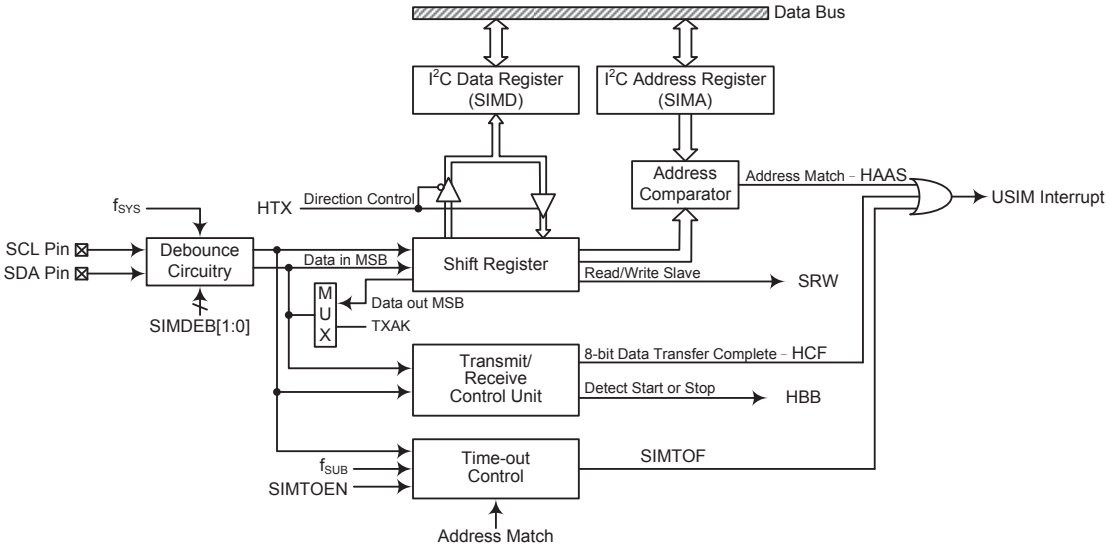


I²C Master Slave Bus Connection

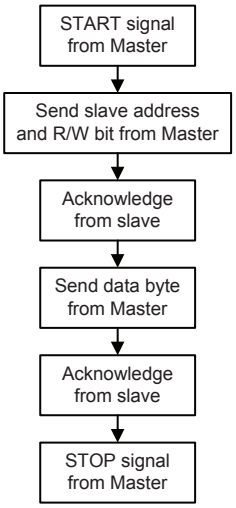
I²C Interface Operation

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

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



I²C Block Diagram



I²C Interface Operation

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

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

I²C Minimum f_{SYS} Frequency Requirements

I²C Registers

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

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

I²C Register List

I²C Data Register

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

- **SIMD Register**

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

"x": unknown

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

I²C Address Register

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

- **SIMA Register**

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

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

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

I²C Control Registers

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

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is CTM0 CCRP match frequency/2
101: SPI slave mode

110: I²C slave mode
111: Unused mode

When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from CTM0 and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

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

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

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

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

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

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

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

- **SIMC1 Register**

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

Bit 7 **HCF**: I²C Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer

The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

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

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

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

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

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

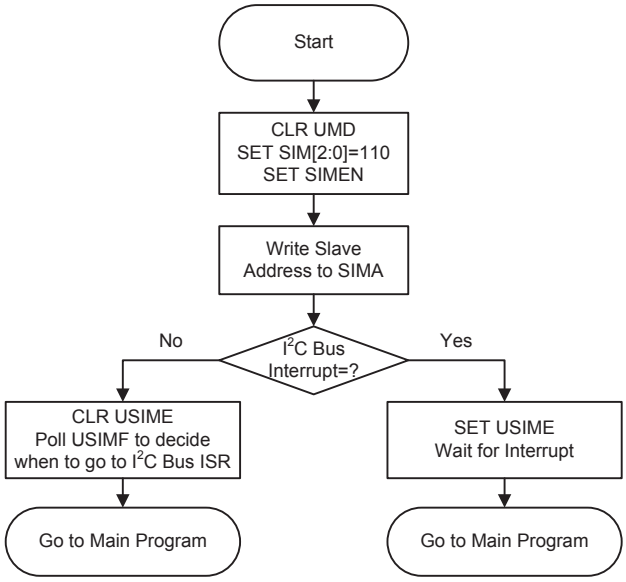
Bit 1 **IAMWU**: I²C Address Match Wake-up control
0: Disable
1: Enable
This bit should be set to 1 to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake up, then this bit must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

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

I²C Bus Communication

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

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



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

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

I²C Slave Address

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

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

I²C Bus Read/Write Signal

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

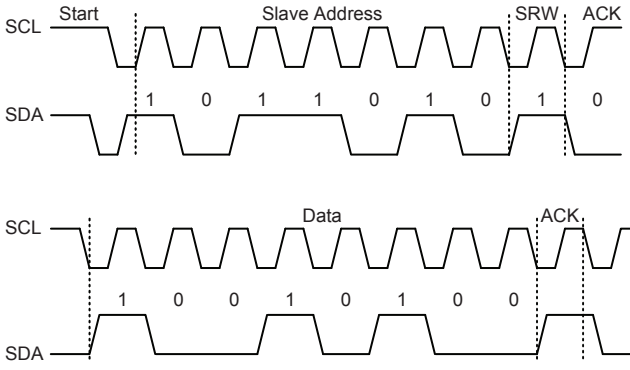
I²C Bus Slave Address Acknowledge Signal

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

I²C Bus Data and Acknowledge Signal

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

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

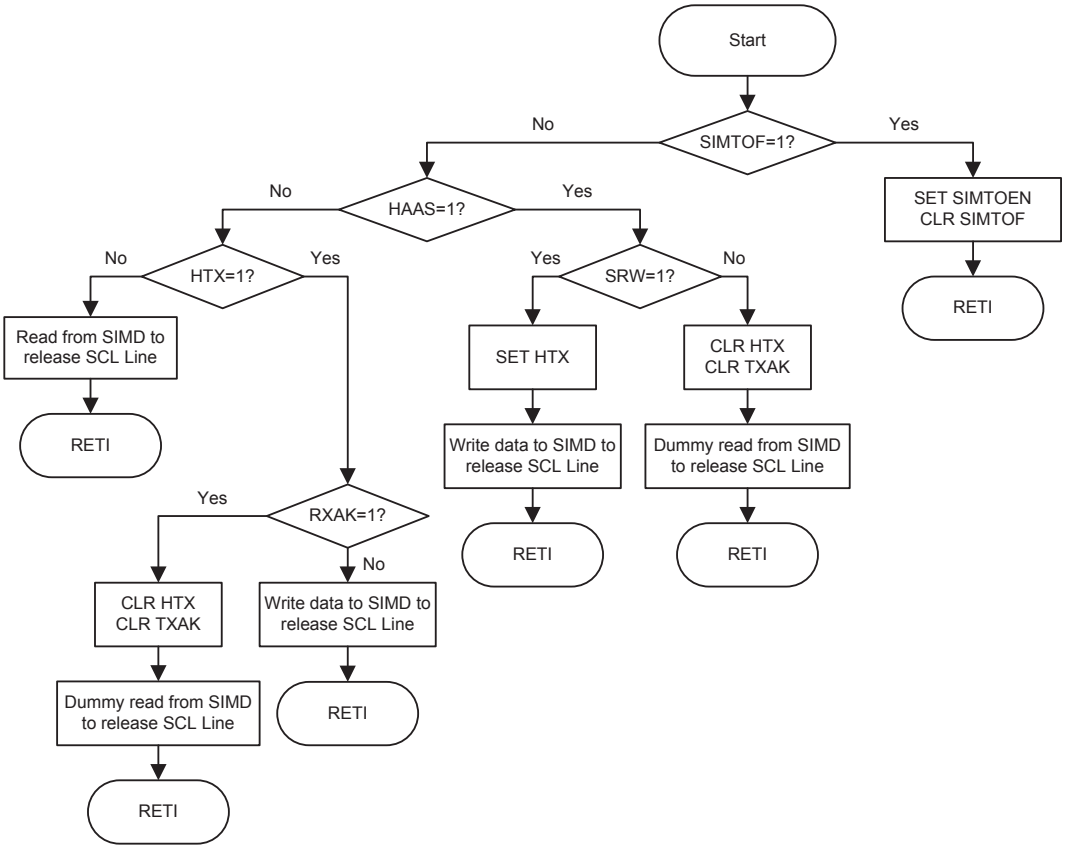


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

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

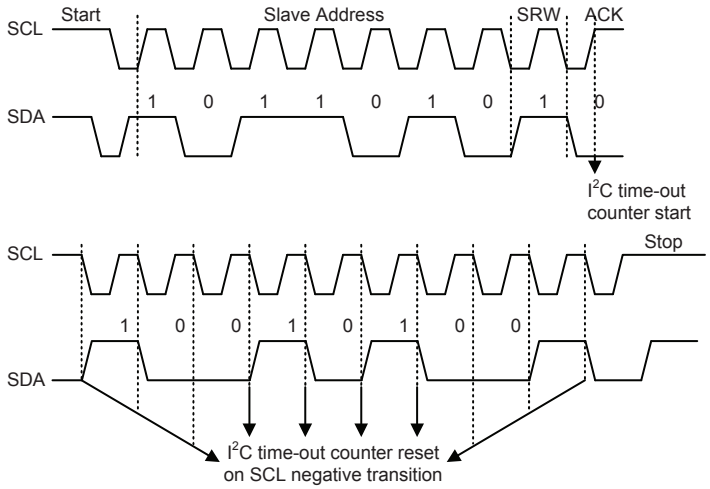
I²C Communication Timing Diagram



I²C Bus ISR Flow Chart

I²C Time-out Control

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



I²C Time-out

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

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

I²C Registers after Time-out

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

- **SIMTOC Register**

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

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

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

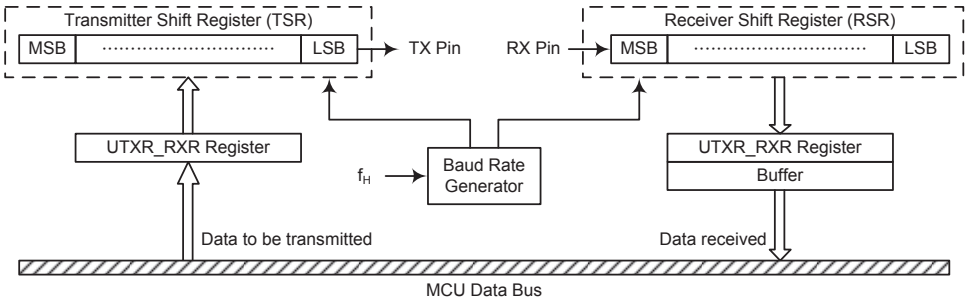
Bit 5~0 **SIMTOS5~SIMTOS0**: USIM I²C Time-out period selection
I²C Time-out clock source is $f_{SUB}/32$
I²C time-out time is equal to (SIMTOS[5:0]+1) × $(32/f_{SUB})$.

UART Interface

The device contains an integrated full-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function shares the same internal interrupt vector with the SPI and I²C interfaces which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UART function contains the following features:

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



UART Data Transfer Block Diagram

UART External Pins

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

UART Data Transfer Scheme

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

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

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

UART Status and Control Registers

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

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

UART Register List

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C Operating Mode Control
When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM function. Refer to the SPI or I²C register section for more details.

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

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

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

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

- **UUSR Register**

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

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

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

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

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

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

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

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

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

Bit 0 **UTXIF**: Transmit UTXR_RXR data register status
0: Character is not transferred to the transmit shift register
1: Character has transferred to the transmit shift register (UTXR_RXR data register is empty)
The UTXIF flag is the transmit data register empty flag. When this read only flag is "0", it indicates that the character is not transferred to the transmitter shift register. When the flag is "1", it indicates that the transmitter shift register has received a

character from the UTXR_RXR data register. The UTXIF flag is cleared to 0 by reading the UART status register (UUSR) with UTXIF set and then writing to the UTXR_RXR data register. Note that when the UTXEN bit is set, the UTXIF flag bit will also be set since the transmit data register is not yet full.

- **UUCR1 Register**

The UUCR1 register together with the UUCR2 register are the two UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

"x": unknown

Bit 7 **UREN**: UART function enable control
0: Disable UART. TX and RX pins are in a floating state
1: Enable UART. TX and RX pins function as UART pins
The UREN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX pin as well as the TX pin will be set in a floating state. When the bit is equal to "1", the UART will be enabled if the UMD bit is set and the TX and RX pins will function as defined by the UTXEN and URXEN enable control bits.
When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF bits will be cleared to 0, while the UTIDLE, UTXIF and URIDLE bits will be set. Other control bits in UUCR1, UUCR2 and UBRG registers will remain unaffected. If the UART is active and the UREN bit is cleared to 0, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6 **UBNO**: Number of data transfer bits selection
0: 8-bit data transfer
1: 9-bit data transfer
This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to "1", a 9-bit data length format will be selected. If the bit is equal to "0", then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits URX8 and UTX8 will be used to store the 9th bit of the received and transmitted data respectively.

Bit 5 **UPREN**: Parity function enable control
0: Parity function is disabled
1: Parity function is enabled
This is the parity enable bit. When this bit is equal to "1", the parity function will be enabled. If the bit is equal to "0", then the parity function will be disabled.

Bit 4 **UPRT**: Parity type selection bit
0: Even parity for parity generator
1: Odd parity for parity generator
This bit is the parity type selection bit. When this bit is equal to "1", odd parity type will be selected. If the bit is equal to "0", then even parity type will be selected.

Bit 3 **USTOPS**: Number of Stop bits selection
0: One stop bit format is used
1: Two stop bits format is used
This bit determines if one or two stop bits are to be used. When this bit is equal to "1", two stop bits are used. If this bit is equal to "0", then only one stop bit is used.

Bit 2 **UTXBRK**: Transmit break character
0: No break character is transmitted
1: Break characters transmit
The UTXBRK bit is the Transmit Break Character bit. When this bit is "0", there are no break characters and the TX pin operates normally. When the bit is "1", there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to "1", after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the UTXBRK bit is reset.

Bit 1 **URX8**: Receive data bit 8 for 9-bit data transfer format (read only)
This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as URX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **UTX8**: Transmit data bit 8 for 9-bit data transfer format (write only)
This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as UTX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

- **UUCR2 Register**

The UUCR2 register is the second of the two UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UART Transmitter and Receiver as well as enabling the various USIM UART mode interrupt sources. The register also serves to control the baud rate speed, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

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

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

Bit 5 **UBRGH**: Baud Rate speed selection
0: Low speed baud rate
1: High speed baud rate

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

Bit 4 **UADDEN**: Address detect function enable control
0: Address detect function is disabled
1: Address detect function is enabled

The bit named UADDEN is the address detect function enable control bit. When this bit is equal to "1", the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to URX7 if UBNO=0 or the 9th bit, which corresponds to URX8 if UBNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of UBNO. If the address bit known as the 8th or 9th bit of the received word is "0" with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **UWAKE**: RX pin wake-up UART function enable control
0: RX pin wake-up UART function is disabled
1: RX pin wake-up UART function is enabled

This bit is used to control the wake-up UART function when a falling edge on the RX pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no RX pin wake-up UART function if the UART clock (f_H) exists. If the UWAKE bit is set to 1 as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the RX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the RX pin when the UWAKE bit is cleared to 0.

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

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

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

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

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

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

- **UTXR_RXR Register**

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

Bit	7	6	5	4	3	2	1	0
Name	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": Unknown

Bit 7~0 **UTXRX7~UTXRX0**: UART Transmit/Receive Data bit 7 ~ bit 0

- **UBRG Register**

Bit	7	6	5	4	3	2	1	0
Name	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": Unknown

Bit 7~0 **UBRG7~UBRG0**: Baud Rate values

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

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

Baud rate= f_H / [16 × (N+1)] if UBRGH=1.

Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the baud rate register UBRG and the second is the value of the UBRGH bit with the control register UUCR2. The UBRGH bit decides if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value N in the UBRG register which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the UBRG register and has a range of between 0 and 255.

UUCR2 UBRGH Bit	0	1
Baud Rate (BR)	$\frac{f_H}{[64(N+1)]}$	$\frac{f_H}{[16(N+1)]}$

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

Calculating the Baud Rate and Error Values

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

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

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

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

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

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

UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding UBNO, UPRT, UPREN, and USTOPS bits in the UUCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

The basic on/off function of the internal UART function is controlled using the UREN bit in the UUCR1 register. When the UART mode is selected by setting the UMD bit in the SIMC0 register to "1", if the UREN, UTXEN and URXEN bits are set, then these two UART pins will act as normal TX output pin and RX input pin respectively. If no data is being transmitted on the TX pin, then it will default to a logic high value.

Clearing the UREN bit will disable the TX and RX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF being cleared while bits UTIDLE, UTXIF and URIDLE will be set. The remaining control bits in the UUCR1, UUCR2 and UBRG registers will remain unaffected. If the UREN bit in the UUCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

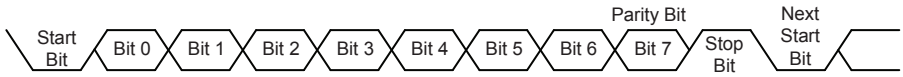
Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UUCR1 register. The UBNO bit controls the number of data bits which can be set to either 8 or 9, the UPRT bit controls the choice of odd or even parity, the UPREN bit controls the parity on/off function and the USTOPS bit decides whether one or two stop bits are to be used. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only used for the transmitter. There is only one stop bit for the receiver.

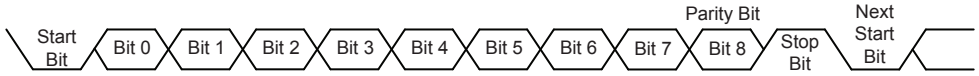
Start Bit	Data Bits	Address Bits	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
Example of 9-bit Data Formats				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

Transmitter Receiver Data Format

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



8-bit data format



9-bit data format

UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the UBNO bit in the UUCR1 register. When UBNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the UTX8 bit in the UUCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the UTXR_RXR register. The data to be transmitted is loaded into this UTXR_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the UTXR_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the UTXEN bit is set, but the data will not be transmitted until the UTXR_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the UTXR_RXR register, after which the UTXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the UTXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the UTXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

Transmitting Data

When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant bit first. In the transmit mode, the UTXR_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the UTX8 bit in the UUCR1 register. The steps to initiate a data transfer can be summarized as follows:

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

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

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

1. A UUSR register access

2. A UTXR_RXR register write execution

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

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

1. A UUSR register access

2. A UTXR_RXR register write execution

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

Transmitting Break

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

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the UBNO bit is set, the word length will be set to 9 bits with the MSB being stored in the URX8 bit of the UUCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

Receiving Data

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

- Make the correct selection of UBNO, UPRT and UPREN bits to define the word length, parity type.
- Setup the UBRG register to select the desired baud rate.
- Set the URXEN bit to ensure that the RX pin is used as a UART receiver pin.

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

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

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

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

1. A UUSR register access
2. A UTXR_RXR register read execution

Receiving Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the UBNO bit plus one stop bit. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by UBNO plus one stop bit. The URXIF bit is set, UFERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the URIDLE bit is set. A break is regarded as a character that contains only zeros with the UFERR flag set. If a long break signal has been detected, the receiver will regard it as a data frame including a start bit, data bits and the invalid stop bit and the UFERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will

not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until stop bits are received. It should be noted that the URIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

- The framing error flag, UFERR, will be set
- The receive data register, UTXR_RXR, will be cleared
- The UOERR, UNF, UPERR, URIDLE or URXIF flags will possibly be set

Idle Status

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

Receiver Interrupt

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

Managing Receiver Errors

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

Overrun Error – UOERR

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

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

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

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

Noise Error – UNF

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

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

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

Framing Error – UFERR

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

Parity Error – UPERR

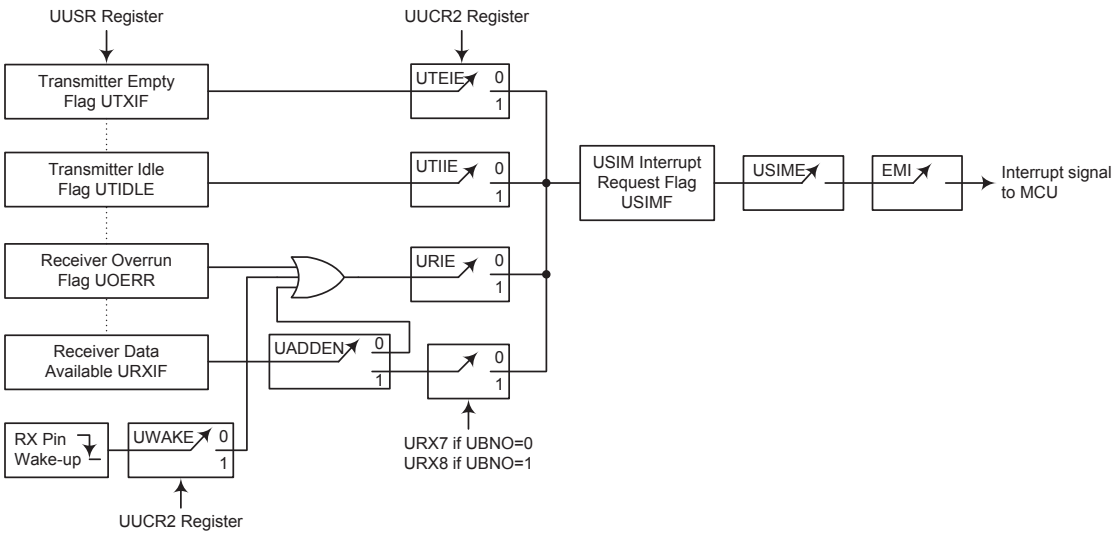
The read only parity error flag, UPERR, in the UUSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, UPREN = 1, and if the parity type, odd or even is selected. The read only UPERR flag and the received data will be recorded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

UART Interrupt Structure

Several individual UART conditions can trigger an USIM interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and the USIM interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding UUSR register flags which will generate an USIM interrupt if its associated interrupt enable control bit in the UUCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual USIM UART mode interrupt sources.

The address detect condition, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt when an address detect condition occurs if its function is enabled by setting the UADDEN bit in the UUCR2 register. An RX pin wake-up, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt if the UART clock (f_H) source is switched off and the UWAKE and URIE bits in the UUCR2 register are set when a falling edge on the RX pin occurs. Note that in the event of an RX wake-up interrupt occurring, there will be a certain period of delay, commonly known as the System Start-up Time, for the oscillator to restart and stabilize before the system resumes normal operation.

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



UART Interface Interrupt Structure

Address Detect Mode

Setting the Address Detect Mode bit, UADDEN, in the UUCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the URXIF flag. If the UADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the USIME and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if UBNO=1 or the 8th bit if UBNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the UADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the URXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit UPREN to zero.

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

UADDEN Bit Function

UART Power Down and Wake-up

When the UART clock (f_H) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock (f_H) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the IDLE or SLEEP Mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP Mode, note that the UUSR, UUCR1, UUCR2, transmit and receive registers, as well as the UBRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the UWAKE bit in the UUCR2 register. If this bit, along with the UART mode selection bit, UMD, the UART enable bit, UREN, the receiver enable bit, URXEN and the receiver interrupt bit, URIE, are all set when the UART clock (f_H) is off, then a falling edge on the RX pin will trigger an RX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX pin will be ignored.

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

Serial Interface – SPIA

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

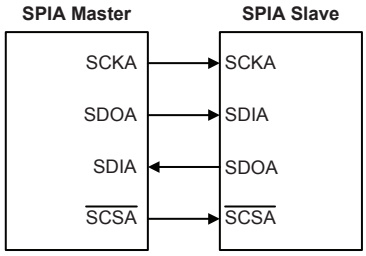
This SPIA interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices, etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying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

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

SPIA Interface Operation

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

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

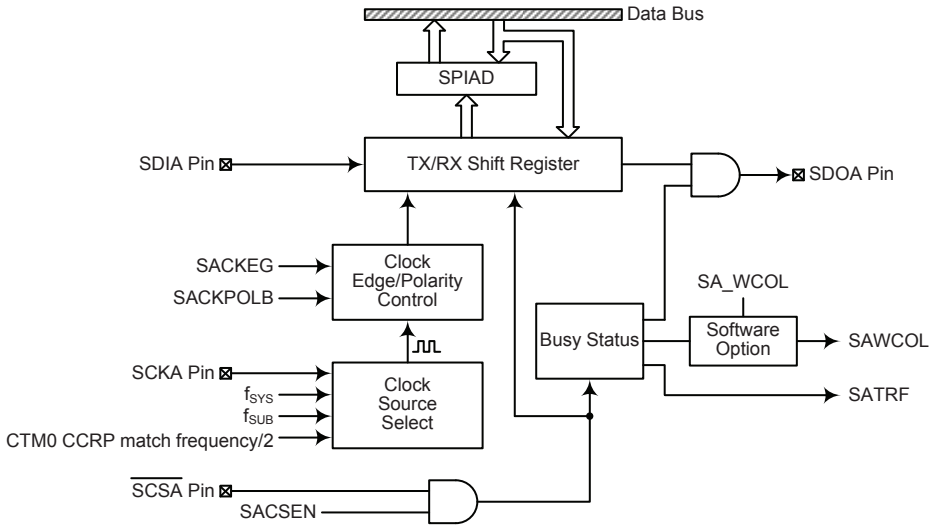


SPIA Master/Slave Connection

The SPIA Serial Interface function includes the following features:

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

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



SPIA Block Diagram

- **SPIA Registers**

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

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

SPIA Register List

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

- **SPIAD Register**

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

"x": unknown

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

- **SPIAC0 Register**

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

Bit 7~5 **SASPI2~SASPI0**: SPIA Operating Mode Control
000: SPIA master mode with clock $f_{SYS}/4$
001: SPIA master mode with clock $f_{SYS}/16$
010: SPIA master mode with clock $f_{SYS}/64$
011: SPIA master mode with clock f_{SUB}
100: SPIA master mode with clock CTM0 CCRP match frequency/2
101: SPIA slave mode
11x: SPIA disable

Bit 4~2 Unimplemented, read as "0"

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

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

- **SPIAC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SACKPOLB	SACKEG	SAMLS	SACSEN	SAWCOL	SATRF
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

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

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

Bit 3 **SAMLS**: SPIA data shift order
0: LSB first
1: MSB first
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2 **SACSEN**: SPIA $\overline{\text{SCSA}}$ pin control
0: Disable
1: Enable
The SACSEN bit is used as an enable/disable for the $\overline{\text{SCSA}}$ pin. If this bit is low, then the $\overline{\text{SCSA}}$ pin function will be disabled and can be placed into I/O pin or other pin-shared functions. If the bit is high, the $\overline{\text{SCSA}}$ pin will be enabled and used as a select pin.

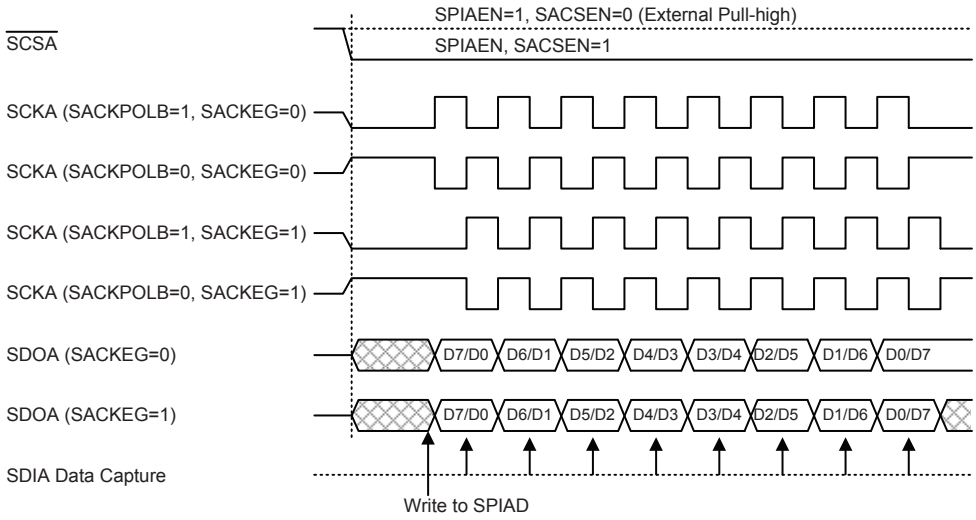
Bit 1 **SAWCOL**: SPIA write collision flag
0: No collision
1: Collision
The SAWCOL flag is used to detect whether a data collision has occurred or not. If this bit is high, it means that data has been attempted to be written to the SPIAD register during a data transfer operation. This writing operation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

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

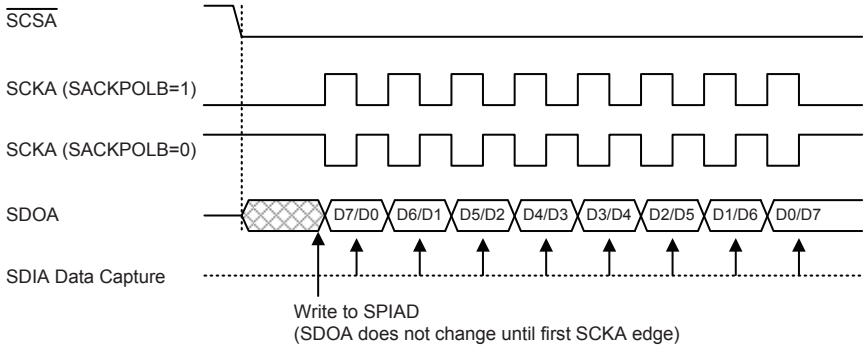
SPIA Communication

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

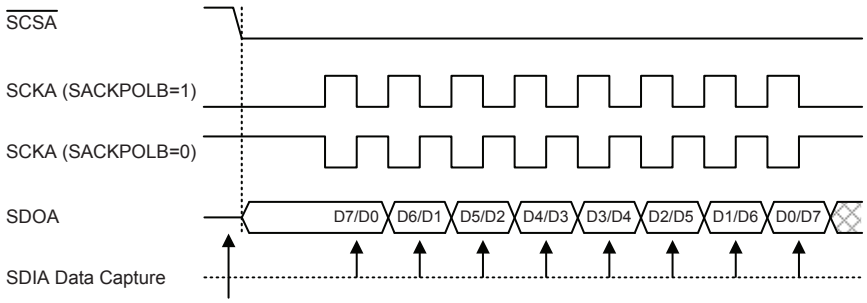
The master should output a $\overline{SCSA}$ signal to enable the slave device before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the SCKA signal depending upon the configurations of the SACKPOLB bit and SACKEG bit. The accompanying timing diagram shows the relationship between the slave data and SCKA signal for various configurations of the SACKPOLB and SACKEG bits. The SPIA will continue to function if the SPIA clock source is active.



SPIA Master Mode Timing

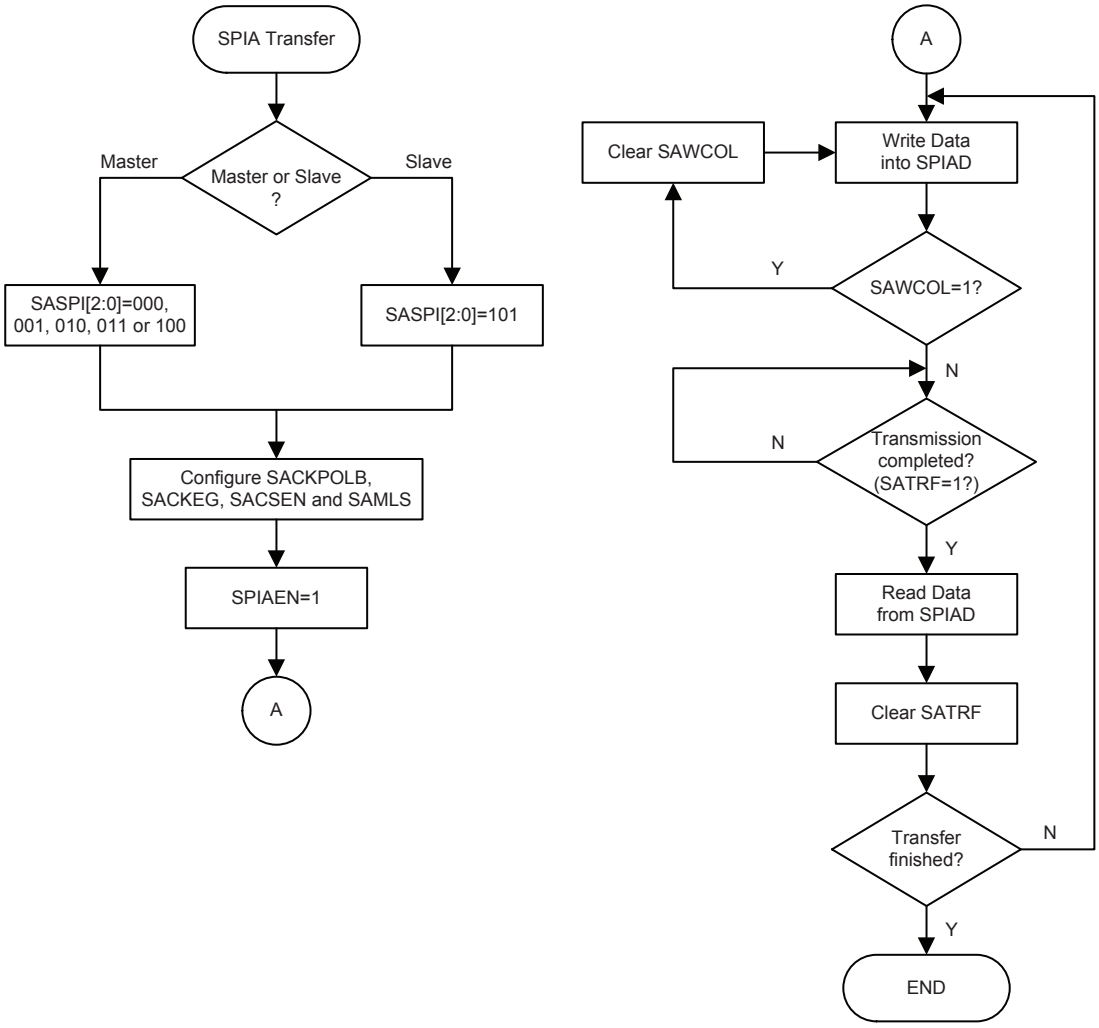


SPIA Slave Mode Timing – SACKEG=0



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

SPIA Slave Mode Timing – SACKEG=1



SPIA Transfer Control Flow Chart

SPIA Bus Enable/Disable

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

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

SPIA Operation

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

The SACSEN bit in the SPIAC1 register controls the overall function of the SPIA interface. Setting this bit high will enable the SPIA interface by allowing the $\overline{SCSA}$ line to be active, which can then be used to control the SPIA interface. If the SACSEN bit is low, the SPIA interface will be disabled and the $\overline{SCSA}$ line will be an I/O pin or other pin-shared functions and can therefore not be used for control of the SPIA interface. If the SACSEN bit and the SPIAEN bit in the SPIAC0 register are set high, this will place the SDIA line in a floating condition and the SDOA line high. If in Master Mode the SCKA line will be either high or low depending upon the clock polarity selection bit SACKPOLB in the SPIAC1 register. If in Slave Mode the SCKA line will be in a floating condition. If SPIAEN is low, then the bus will be disabled and $\overline{SCSA}$, SDIA, SDOA and SCKA pins will all become I/O pins or other pin-shared functions using the corresponding pin-shared function selection bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SPIAD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode

- Step 1
  Select the clock source and Master mode using the SASPI2~SASPI0 bits in the SPIAC0 control register.
- Step 2
  Setup the SACSEN bit and setup the SAMLS bit to choose if the data is MSB or LSB shifted first, this must be same as the Slave device.
- Step 3
  Setup the SPIAEN bit in the SPIAC0 control register to enable the SPIA interface.
- Step 4
  For write operations: write the data to the SPIAD register, which will actually place the data into the TXRX buffer. Then use the SCKA and $\overline{SCSA}$ lines to output the data. After this go to step 5.
  For read operations: the data transferred in on the SDIA line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPIAD register.
- Step 5
  Check the SAWCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.

- Step 6
  Check the SATRF bit or wait for a SPIA serial bus interrupt.
- Step 7
  Read data from the SPIAD register.
- Step 8
  Clear SATRF.
- Step 9
  Go to step 4.

Slave Mode

- Step 1
  Select the SPI Slave mode using the SASPI2~SASPI0 bits in the SPIAC0 control register
- Step 2
  Setup the SACSEN bit and setup the SAMLS bit to choose if the data is MSB or LSB shifted first, this setting must be the same with the Master device.
- Step 3
  Setup the SPIAEN bit in the SPIAC0 control register to enable the SPIA interface.
- Step 4
  For write operations: write the data to the SPIAD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCKA and $\overline{\text{SCSA}}$ signal. After this, go to step 5.
  For read operations: the data transferred in on the SDIA line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPIAD register.
- Step 5
  Check the SAWCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the SATRF bit or wait for a SPIA serial bus interrupt.
- Step 7
  Read data from the SPIAD register.
- Step 8
  Clear SATRF.
- Step 9
  Go to step 4.

Error Detection

The SAWCOL bit in the SPIAC1 register is provided to indicate errors during data transfer. The bit is set by the SPIA serial Interface but must be cleared by the application program. This bit indicates a data collision has occurred which happens if a write to the SPIAD register takes place during a data transfer operation and will prevent the write operation from continuing.

LCD Driver

For large volume applications, which incorporate an LCD in their design, the use of a custom display rather than a more expensive character based display reduces costs significantly. However, the corresponding COM and SEG signals required, which vary in both amplitude and time, to drive such a custom display require many special considerations for proper LCD operation to occur. This 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	Wave Type
32×4	1/4	1/3	C	A or B

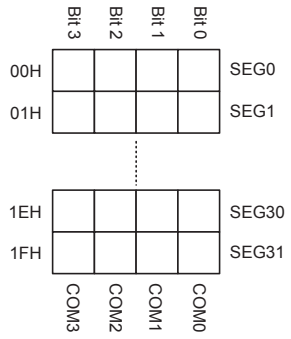
Note: The SEG30~SEG31 pins can be used for the OCDS EV chip and special package only.

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 Data 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.



Note: The SEG30~SEG31 pins can be used for the OCDS EV chip and special package only.

LCD Memory Map

LCD Clock Source

The LCD clock source is the internal clock signal, f_{LCD}, which is sourced from the LXT oscillator output frequency of $f_{LXT}/8$. For proper LCD operation, this arrangement is provided to generate an ideal LCD clock source frequency of 4kHz.

C-type LCD Pump Clock Source

The C-type LCD pump clock source, f_{LCDP}, is provided by the external LXT oscillator clock source f_{LXT}, which is divided by a factor of 1, 2, 4 or 8 using an internal divider circuit. The actual division factor is determined using the LCDPCK2~LCDPCK0 bits in the LCDC register.

LCDPCK[2:0] Bits	f_{LCDP}	Frequency
000~011	$f_{LXT}/8$	4kHz
100	$f_{LXT}/4$	8kHz
101	$f_{LXT}/2$	16kHz
110~111	f_{LXT}	32kHz

LCD Register

The control register LCDC is used to control the various setup features of the LCD Driver.

Various bits in this registers control functions such as LCD wave type, LCD power source and LCD pump clock source selection together with the overall LCD enable/disable control. The LCDEN bit in the LCDC register, which provides the overall LCD enable/disable function. The TYPE bit in the LCDC register is used to select whether Type A or Type B LCD waveform signals are used. Bits LCDP1 and LCDP0 in the LCDC register are used to select the power source to supply the C type LCD panel with the correct bias voltages. The LCDPCK2~LCDPCK0 bits are used to determined the C-type LCD pump clock.

- **LCDC Register**

Bit	7	6	5	4	3	2	1	0
Name	TYPE	—	LCDP1	LCDP0	LCDPCK2	LCDPCK1	LCDPCK0	LCDEN
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

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

Bit 6 Unimplemented, read as "0"

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

Bit 3~1 **LCDPCK2 ~LCDPCK0**: C-type LCD Pump Clock divider
000: 250Hz ($f_{LCDP}/16$)
001: 500Hz ($f_{LCDP}/8$)
010: 1kHz ($f_{LCDP}/4$)
011: 2kHz ($f_{LCDP}/2$)
100: 4kHz ($f_{LCDP}/2$)
101: 8kHz ($f_{LCDP}/2$)
110: 16kHz ($f_{LCDP}/2$)
111: 16kHz ($f_{LCDP}/2$)

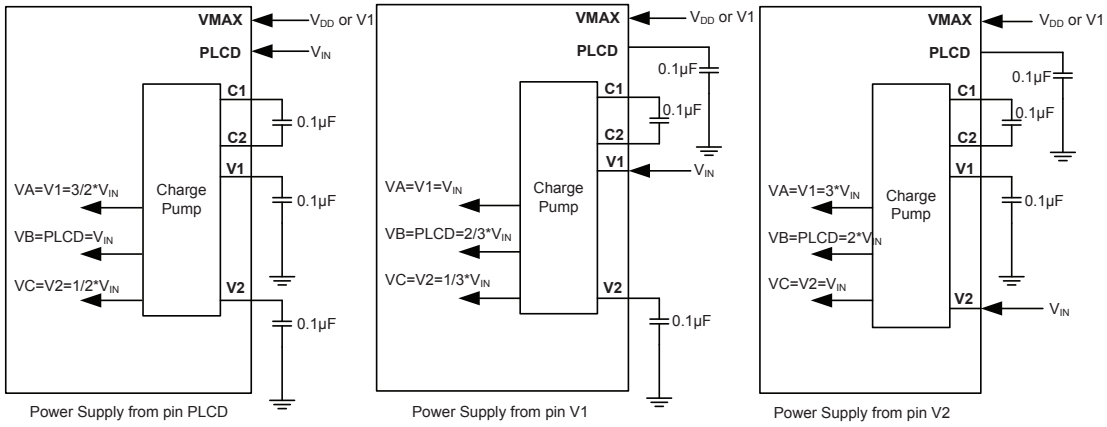
Note: These frequency options are figured out based on the LCD pump clock frequency f_{LCDP}. The actual f_{LCDP} frequency is determined by the LCDPCK2 ~LCDPCK0 bits configurations, which are listed in the "C-type LCD Pump Clock Source" section.

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

LCD Voltage Source and Biasing

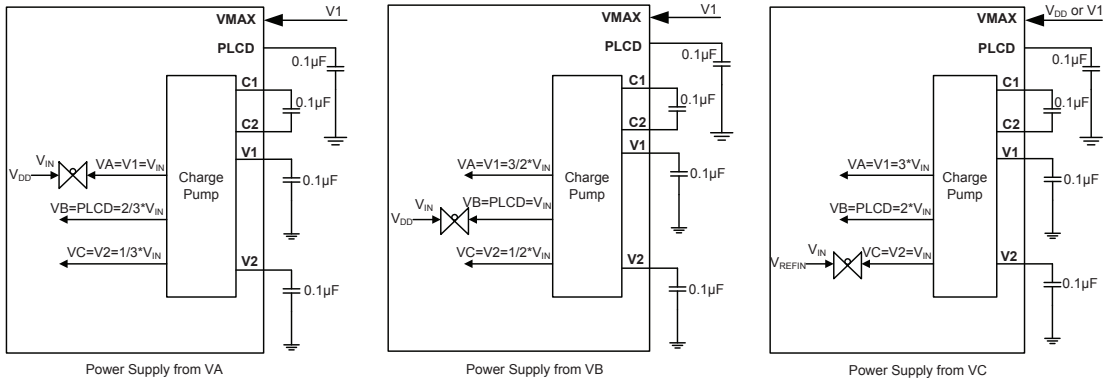
For C type biasing the LCD voltage source can be supplied on the external pin PLCD, V1 or V2 or derived from the internal voltage source to generate the required biasing voltages. The C type bias voltage source is selected using the LCDP1 and LCDP0 bits in the LCDC register. The C type biasing scheme uses an internal charge pump circuit and can generate voltages higher than what is supplied on PLCD or V2. This feature is useful in applications where the microcontroller supply voltage is less than the supply voltage required by the LCD. Additional charge pump capacitors must also be connected between pins C1 and C2 to generate the necessary voltage levels.

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



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

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



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

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

LCD Power Supply		VA Voltage	VB Voltage	VC Voltage
External Power Supply	V_{IN} on V1	V_{IN}	2/3 × V_{IN}	1/3 × V_{IN}
	V_{IN} on PLCD	3/2 × V_{IN}	V_{IN}	1/2 × V_{IN}
	V_{IN} on V2	3 × V_{IN}	2 × V_{IN}	V_{IN}
Internal Power Supply	V_{DD} on VA	V_{DD}	2/3 × V_{DD}	1/3 × V_{DD}
	V_{DD} on VB	3/2 × V_{DD}	V_{DD}	1/2 × V_{DD}
	V_{REFIN} on VC	3 × V_{REFIN}	2 × V_{REFIN}	V_{REFIN}

C Type Bias Power Supply Scheme

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

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

C Type Bias VMAX Pin Connection

LCD Reset Status

The LCD has an internal reset function. Clearing the LCDEN bit to zero will reset the LCD 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	LCDEN	LCD Reset	COM & SEG Voltage Level
No	1	No	Normal Operation
No	0	Yes	Low
Yes	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

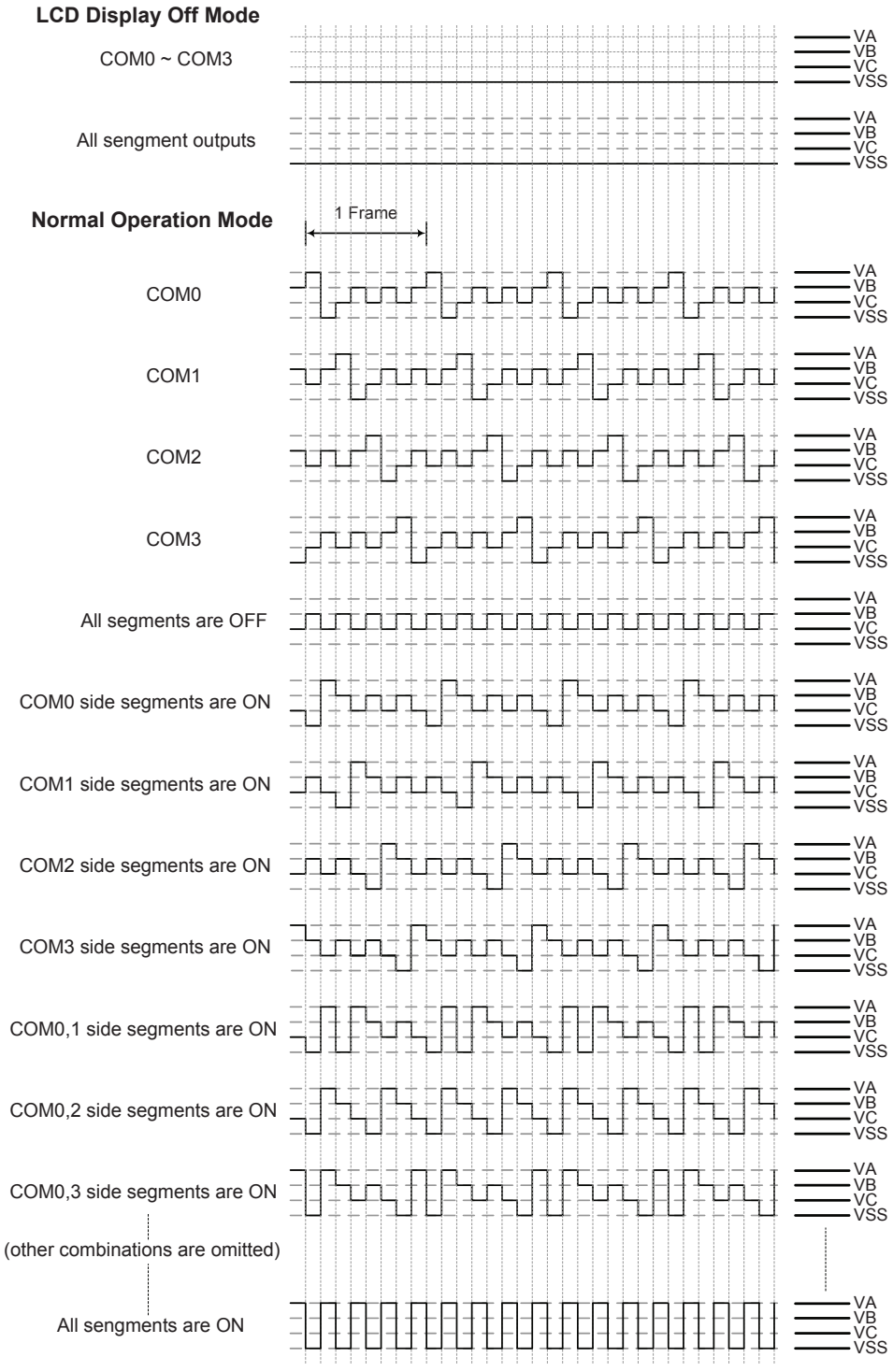
LCD Driver Output

The output structure of the LCD driver is 32×4. The LCD driver bias type has C type only and has a fixed bias of 1/3. Note that the SEG30~SEG31 pins can be used for the OCDS EV chip and special package only.

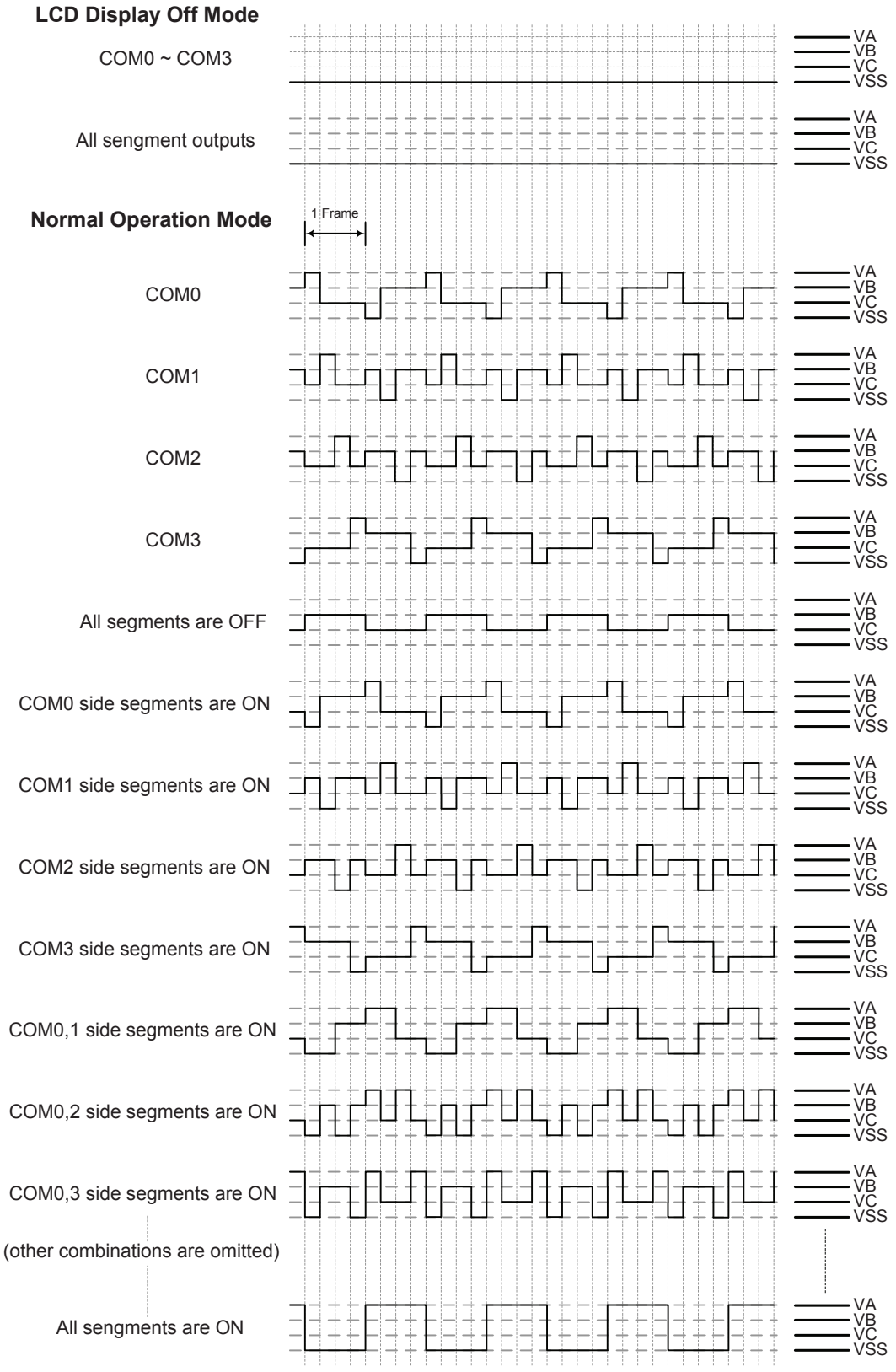
The nature of Liquid Crystal Displays require that only AC voltages can be applied to their pixels as the application of DC voltages to LCD pixels may cause permanent damage. For this reason the relative contrast of an LCD display is controlled by the actual RMS voltage applied to each pixel, which is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

The requirement to limit the DC voltage to zero and to control as many pixels as possible with a minimum number of connections requires that both a time and amplitude signal is generated and applied to the application LCD. These time and amplitude varying signals are automatically generated by the LCD driver circuits in the microcontroller. What is known as the duty determines the number of common lines used, which are also known as backplanes or COMs. The duty, which 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 LCDC register. Type B offers lower frequency signals, however, lower frequencies may introduce flickering and influence display clarity.



LCD Driver Output – Type A, 1/4 duty, 1/3 bias



LCD Driver Output – Type B, 1/4 duty, 1/3 bias

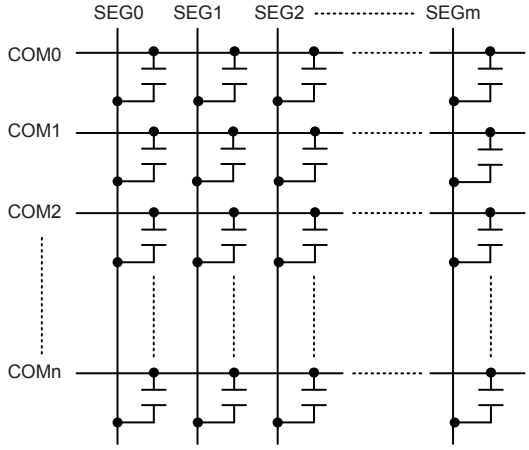
Programming Considerations

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 LCDC register permits the display to be powered off to reduce power consumption. If this bit is zero, the driving signals to the display will cease, producing a blank display pattern but reducing any power consumption associated with the LCD.

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



LCD Panel Equivalent Circuit

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module requires microcontroller attention, 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~INT3 pins, while the internal interrupts are generated by various internal functions such as the Timer Modules (TM), Time Bases, Low Voltage Detector (LVD), EEPROM, SPIA and the USIM module.

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	Note
Global	EMI	—	—
INTn Pin	INTnE	INTnF	n=0~3
Multi-function	MFnE	MFnF	n=0~2
Time Base	TBnE	TBnF	n=0~1
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
USIM	USIME	USIMF	—
SPIA	SPIAE	SPIAF	—
TM	CTMnPE	CTMnPF	n=0~1
	CTMnAE	CTMnAF	
	STMPE	STMPF	—
	STMAE	STMAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	USIMF	INT1F	INT0F	USIME	INT1E	INT0E	EMI
INTC1	MF2F	MF1F	MF0F	SPIAF	MF2E	MF1E	MF0E	SPIAE
INTC2	INT3F	INT2F	TB1F	TB0F	INT3E	INT2E	TB1E	TB0E
MFI0	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
MFI1	—	—	STMAF	STMPF	—	—	STMAE	STMPE
MFI2	—	—	DEF	LVF	—	—	DEE	LVE

Interrupt Register List

- **INTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **INT3S1~INT3S0**: Interrupt Edge Control for INT3 Pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

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

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

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

- **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	USIMF	INT1F	INT0F	USIME	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 **USIMF**: USIM 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 **USIME**: USIM 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	MF2F	MF1F	MF0F	SPIAF	MF2E	MF1E	MF0E	SPIAE
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 **MF2F**: Multi-function Interrupt 2 Request Flag
0: No request
1: Interrupt request

Bit 6 **MF1F**: Multi-function Interrupt 1 Request Flag
0: No request
1: Interrupt request

Bit 5 **MF0F**: Multi-function Interrupt 0 Request Flag
0: No request
1: Interrupt request

Bit 4 **SPIAF**: SPIA Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **MF2E**: Multi-function Interrupt 2 Control
0: Disable
1: Enable

Bit 2 **MF1E**: Multi-function Interrupt 1 Control
0: Disable
1: Enable

Bit 1 **MF0E**: Multi-function Interrupt 0 Control
0: Disable
1: Enable

Bit 0 **SPIAE**: SPIA Interrupt Control
0: Disable
1: Enable

- **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	INT3F	INT2F	TB1F	TB0F	INT3E	INT2E	TB1E	TB0E
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 **INT3F**: External Interrupt 3 Request Flag
0: No request
1: Interrupt request

Bit 6 **INT2F**: External Interrupt 2 Request Flag
0: No request
1: Interrupt request

Bit 5 **TB1F**: Time Base 1 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **TB0F**: Time Base 0 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **INT3E**: External Interrupt 3 Control
0: Disable
1: Enable

Bit 2 **INT2E**: External Interrupt 2 Control
0: Disable
1: Enable

Bit 1 **TB1E**: Time Base 1 Interrupt Control
0: Disable
1: Enable

Bit 0 **TB0E**: Time Base 0 Interrupt Control
0: Disable
1: Enable

- **MFI0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CTM1AF**: CTM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **CTM1PF**: CTM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request

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

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

Bit 3 **CTM1AE**: CTM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **CTM1PE**: CTM1 Comparator P match interrupt control
0: Disable
1: Enable

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

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

- **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	STMAF	STMPF	—	—	STMAE	STMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **STMAF**: STM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **STMPF**: STM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **STMAE**: STM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **STMPE**: STM Comparator P match interrupt control
0: Disable
1: Enable

- **MFI2 Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **DEF**: Data EEPROM interrupt request flag
0: No request
1: Interrupt request

Bit 4 **LVF**: LVD interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

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

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

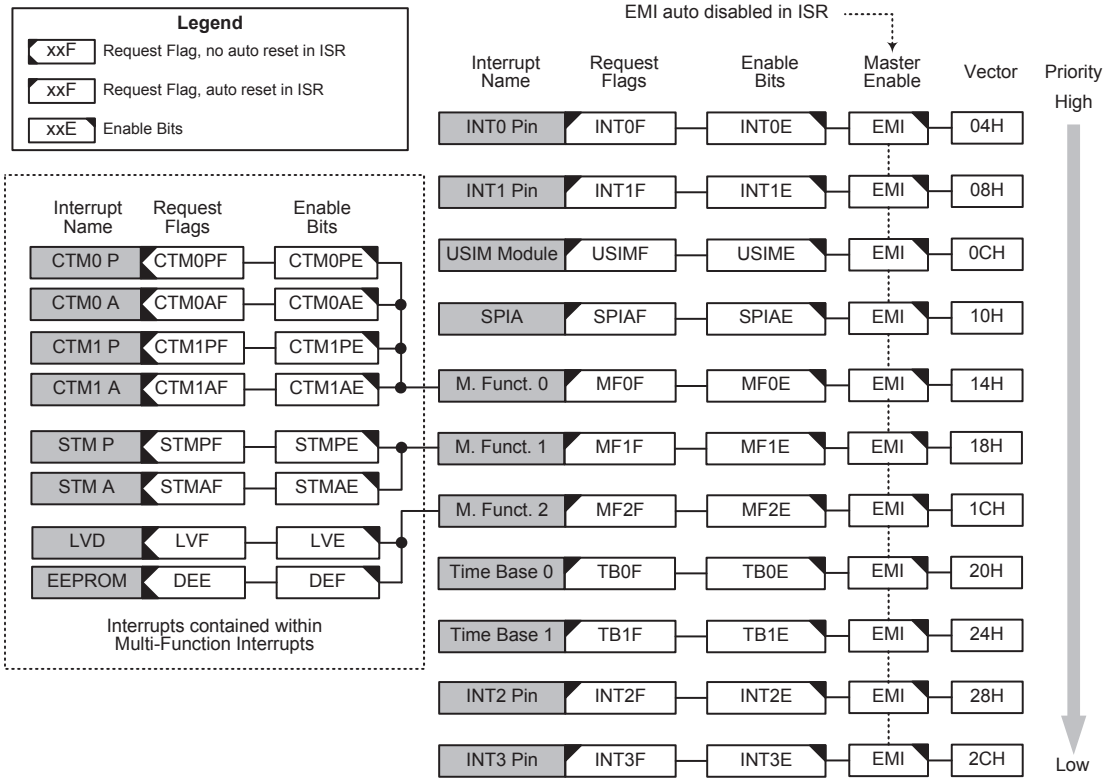
Interrupt Operation

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

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

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

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



Interrupt Structure

External Interrupt

The external interrupts are controlled by signal transitions on the pins INT0~INT3. An external interrupt request will take place when the external interrupt request flags, INT0F~INT3F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT3E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set 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~INT3F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input. The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

Multi-function Interrupt

Within the 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, LVD interrup and EEPROM write operation 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 request flags will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupts will not be automatically reset and must be manually reset by the application program.

Timer Module Interrupts

The Compact and Standard type TMs each has two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. For all of the TM types there are two interrupt request flags and two enable control bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

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

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.

USIM Interrupt

The Universal Serial Interface Module Interrupt, also known as the USIM interrupt, will take place when the USIM Interrupt request flag, USIMF, is set. As the USIM interface can operate in three modes which are SPI mode, I^2C mode and UART mode, the USIMF flag can be set by different conditions depending on the selected interface mode.

If the SPI or I^2C mode is selected, the USIM interrupt can be triggered when a byte of data has been received or transmitted by the USIM SPI or I^2C interface, or an I^2C slave address match occurs, or an I^2C bus time-out occurs. If the UART mode is selected, several individual UART conditions including a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up, can generate a USIM interrupt with the USIMF flag bit set high.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Universal Serial Interface Module Interrupt enable bit, USIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the respective Interrupt vector, will take place. When the interrupt is serviced, the Universal Serial Interface Module Interrupt flag, USIMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Note that if the USIM interrupt is triggered by the UART interface, after the interrupt has been serviced, the UUSR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

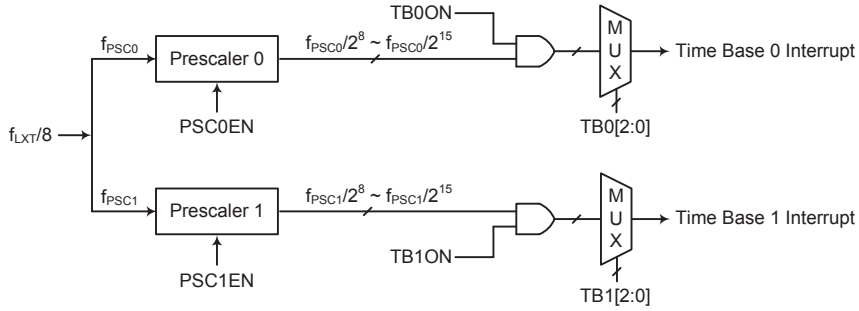
SPIA Interface Interrupt

An SPIA Interrupt request will take place when the SPIA Interrupt request flag, SPIAF, is set, which occurs when a byte of data has been received or transmitted by the SPIA interface. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, SPIAE, must first be set. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SPIA interface, a subroutine call to the respective Interrupt vector, will take place. When the SPIA Interface Interrupt is serviced, the SPIAF flag 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 cleared, the EMI bit will also be automatically 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_{PSC0} or f_{PSC1}, originates from the internal clock source $f_{LXT}/8$ 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.



Time Base Interrupts

- **PSCnR Register (n=0~1)**

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

Bit7 ~ 1 Unimplemented, read as "0"

Bit0 **PSCnEN**: Prescaler n clock enable control
0: Disable
1: Enable
This PSCnEN bit is the Prescaler n clock enable/disable control bit. When the Prescale clock is disabled, it can reduce extra power consumption. Prescaler n clock is sourced from the LXT oscillator output frequency of $f_{LXT}/8$.

- **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: $2^8/f_{PSC0}$
001: $2^9/f_{PSC0}$
010: $2^{10}/f_{PSC0}$
011: $2^{11}/f_{PSC0}$
100: $2^{12}/f_{PSC0}$
101: $2^{13}/f_{PSC0}$
110: $2^{14}/f_{PSC0}$
111: $2^{15}/f_{PSC0}$

- **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: $2^8/f_{PSC1}$
001: $2^9/f_{PSC1}$
010: $2^{10}/f_{PSC1}$
011: $2^{11}/f_{PSC1}$
100: $2^{12}/f_{PSC1}$
101: $2^{13}/f_{PSC1}$
110: $2^{14}/f_{PSC1}$
111: $2^{15}/f_{PSC1}$

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}, 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. Four bits in this register, VLVD3~VLVD0, are used to select one of sixteen fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

- **LVDC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	VLVD3	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"

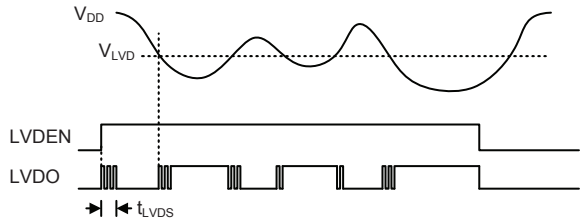
Bit 5 **LVDO**: LVD Output Flag
0: No Low Voltage Detect
1: Low Voltage Detect

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

Bit 3~0 **VLVD3~VLVD0**: Select LVD Voltage
0000: 1.8V
0001: 1.9V
0010: 2.0V
0011: 2.1V
0100: 2.2V
0101: 2.3V
0110: 2.4V
0111: 2.5V
1000: 2.6V
1001: 2.7V
1010: 2.8V
1011: 2.9V
1100: 3.0V
1101: 3.3V
1110: 3.6V
1111: 4.0V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.8V and 4.0V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. When the device is in the SLEEP mode, the low voltage detector will be disabled even if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} falls below the preset LVD voltage. This will cause the device to wake-up from the 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.

Configuration Options

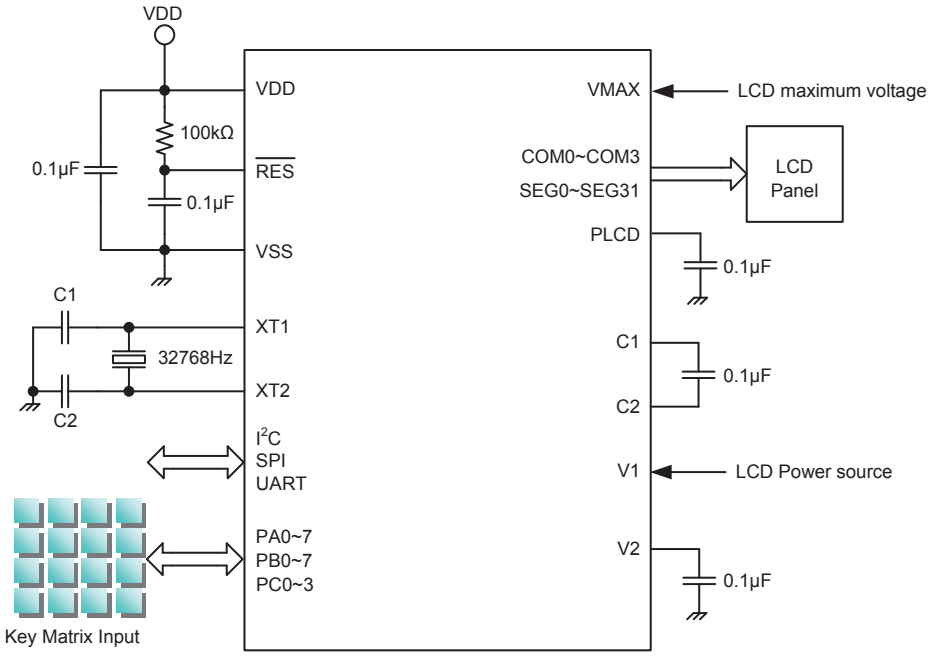
Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later using the application program. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
1	HIRC frequency selection: 1. 4MHz 2. 8MHz 3. 12MHz

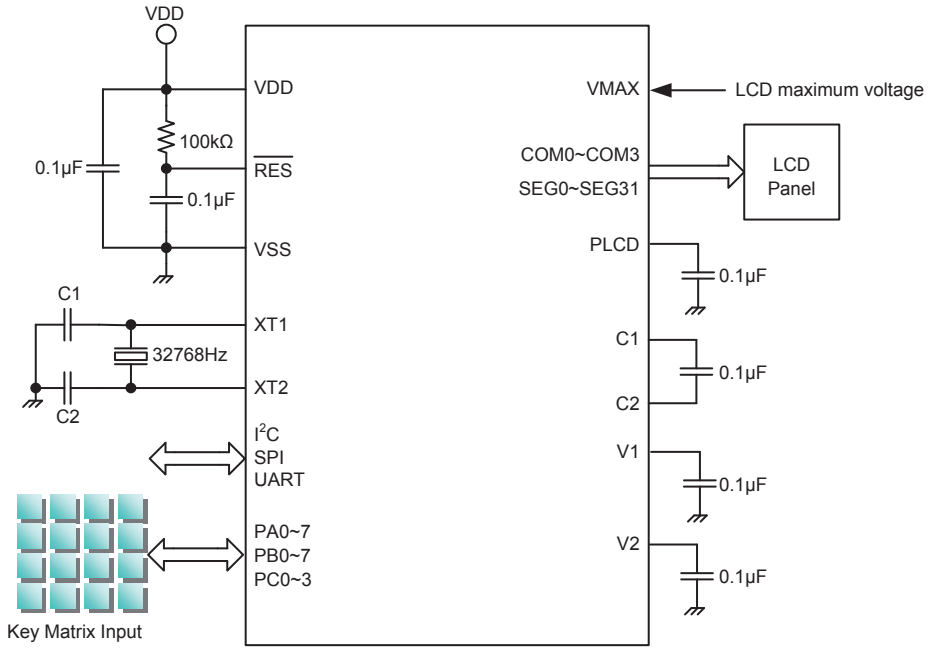
Note: When the HIRC has been configured at a frequency shown in this table, the HIRC1 and HIRC0 bits should also be setup to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Application Circuits

LCD Power Source from External



LCD Power Source from Internal Regulator



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of several kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

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

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

Note: 1. For skip instructions, if the result of the comparison involves a skip then up to three cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the "CLR WDT" instruction the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after the "CLR WDT" instructions is executed. Otherwise the TO and PDF flags remain unchanged.

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sections except sector 0, the extended instruction can be used to access the data memory instead of using the indirect addressing access to improve the CPU firmware performance.

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	2Note	Z, C, AC, OV, SC
LADC A,[m]	Add Data Memory to ACC with Carry	2	Z, C, AC, OV, SC
LADCM A,[m]	Add ACC to Data memory with Carry	2Note	Z, C, AC, OV, SC
LSUB A,[m]	Subtract Data Memory from ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LSBC A,[m]	Subtract Data Memory from ACC with Carry	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	2Note	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2Note	C
Logic Operation			
LAND A,[m]	Logical AND Data Memory to ACC	2	Z
LOR A,[m]	Logical OR Data Memory to ACC	2	Z
LXOR A,[m]	Logical XOR Data Memory to ACC	2	Z
LANDM A,[m]	Logical AND ACC to Data Memory	2Note	Z
LORM A,[m]	Logical OR ACC to Data Memory	2Note	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2Note	Z
LCPL [m]	Complement Data Memory	2Note	Z
LCPLA [m]	Complement Data Memory with result in ACC	2	Z
Increment & Decrement			
LINCA [m]	Increment Data Memory with result in ACC	2	Z
LINC [m]	Increment Data Memory	2Note	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2Note	Z
Rotate			
LRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2Note	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2Note	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2Note	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2Note	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2Note	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2Note	None
LSET [m].i	Set bit of Data Memory	2Note	None

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

Note: 1. For these extended skip instructions, if the result of the comparison involves a skip then up to four cycles are required, if no skip takes place two cycles is required.
2. Any extended instruction which changes the contents of the PCL register will also require three cycles for execution.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry

Description: The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + [m] + C

Affected flag(s): OV, Z, AC, C, SC

ADCM A,[m] Add ACC to Data Memory with Carry

Description: The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation: [m] ← ACC + [m] + C

Affected flag(s): OV, Z, AC, C, SC

ADD A,[m] Add Data Memory to ACC

Description: The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + [m]

Affected flag(s): OV, Z, AC, C, SC

ADD A,x Add immediate data to ACC

Description: The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + x

Affected flag(s): OV, Z, AC, C, SC

ADDM A,[m] Add ACC to Data Memory

Description: The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation: [m] ← ACC + [m]

Affected flag(s): OV, Z, AC, C, SC

AND A,[m] Logical AND Data Memory to ACC

Description: Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation: ACC ← ACC "AND" [m]

Affected flag(s): Z

AND A,x Logical AND immediate data to ACC

Description: Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.

Operation: ACC ← ACC "AND" x

Affected flag(s): Z

ANDM A,[m] Logical AND ACC to Data Memory

Description: Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation: [m] ← ACC "AND" [m]

Affected flag(s): Z

CALL addr Subroutine call

Description Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

DEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

Description This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

OR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" x

Affected flag(s) Z

ORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

RET Return from subroutine

Description The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation Program Counter ← Stack

Affected flag(s) None

RET A,x Return from subroutine and load immediate data to ACC

Description The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

Description The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

RLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

RR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory is rotated right by 1 bit with bit 0
rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s): C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: ACC ← ACC − [m] − $\overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

SBC A, x Subtract immediate data from ACC with Carry

Description: The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: ACC ← ACC - [m] - $\overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: [m] ← ACC − [m] − $\overline{C}$

Affected flag(s): OV, Z, AC, C, SC, CZ

SDZ [m] Skip if decrement Data Memory is 0

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation:
[m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation:
ACC ← [m] − 1
Skip if ACC=0

Affected flag(s): None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

SNZ [m].i Skip if Data Memory is not 0

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

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

Operation Skip if [m]≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − x

Affected flag(s) OV, Z, AC, C, SC, CZ

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

Description The low byte of the program code (specific page) addressed by the table pointer pair (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRD [m] Increment table pointer low byte first and read table to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

XOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Extended Instruction Definition

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

LADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

LADCM A,[m] Add ACC to Data Memory with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C, SC

LADD A,[m] Add Data Memory to ACC

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

LADDM A,[m] Add ACC to Data Memory

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m]

Affected flag(s) OV, Z, AC, C, SC

LAND A,[m] Logical AND Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

LANDM A,[m] Logical AND ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″AND″ [m]

Affected flag(s) Z

LCLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

LCLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

LCPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation $[m] \leftarrow \overline{[m]}$

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow \overline{[m]}$

Affected flag(s) Z

LDAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

LDEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation $[m] \leftarrow [m] - 1$

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

LINC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation $[m] \leftarrow [m] + 1$

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] + 1$

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

LMOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

LOR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

LORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

LRL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

LRLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

LRLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

LRLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

LRR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

LRRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s) None

LRRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

LRRCA [m] Rotate Data Memory right through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s) C

LSBC A,[m] Subtract Data Memory from ACC with Carry

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m] − $\overline{C}$

Affected flag(s) OV, Z, AC, C, SC, CZ

LSBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m] − $\overline{C}$

Affected flag(s) OV, Z, AC, C, SC, CZ

LSDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

LSET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

LSET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

LSIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

LSIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

LSNZ [m].i Skip if Data Memory is not 0

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

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

Operation Skip if [m] ≠ 0

Affected flag(s) None

LSUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

LSUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

LSWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

LSZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

LTABRD [m] Read table (current page) to TBLH and Data Memory

Description The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LTABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRD [m] Increment table pointer low byte first and read table to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LXOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

LXORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

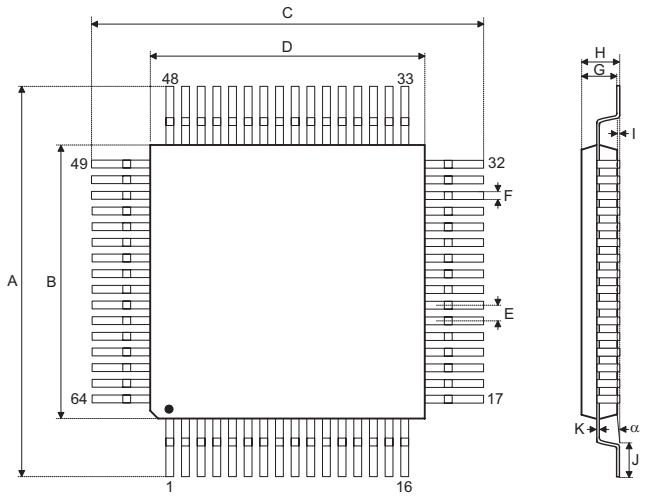
Package Information

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

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

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

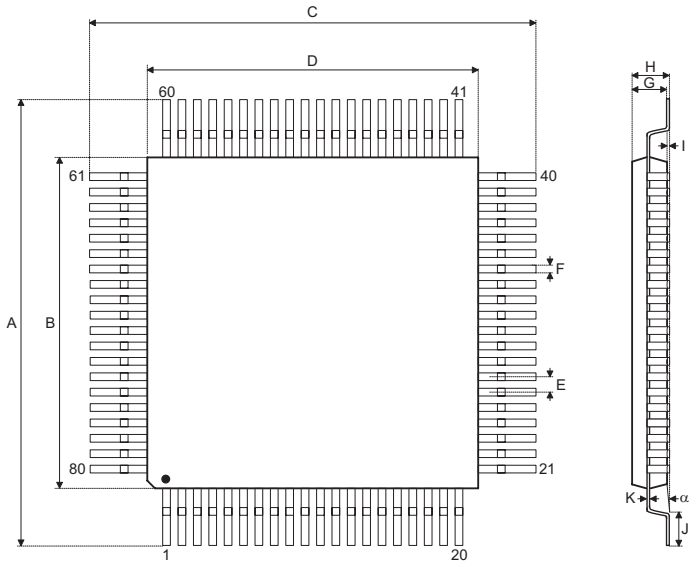
64-pin LQFP (7mm×7mm) Outline Dimensions



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	—	9.0 BSC	—
B	—	7.0 BSC	—
C	—	9.0 BSC	—
D	—	7.0 BSC	—
E	—	0.4 BSC	—
F	0.13	0.18	0.23
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

80-pin LQFP (10mm×10mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.472 BSC	—
B	—	0.394 BSC	—
C	—	0.472 BSC	—
D	—	0.394 BSC	—
E	—	0.015 BSC	—
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	12.00 BSC	—
B	—	10.00 BSC	—
C	—	12.00 BSC	—
D	—	10.00 BSC	—
E	—	0.40 BSC	—
F	0.13	0.18	0.23
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°