V2/24

NFC Reader Touch Flash MCU

BS65F2042

Singel 3 | B-2550 Kontich | Belgium | Tel. +32 (0)3 458 30 33
info@alcom.be | www.alcom.be
Rivium 1e straat 52 | 2909 LE Capelle aan den IJssel | The Netherlands
Tel. +31 (0)10 288 25 00 | info@alcom.nl | www.alcom.nl

Revision: V1.00 Date: January 29, 2024

www.holtek.com

Table of Contents

Features

CPU Features

- Operating voltage
  - ♦ f_{SYS}=8MHz: 1.8V~5.5V
  - ♦ f_{SYS}=12MHz: 2.7V~5.5V
  - ♦ f_{SYS}=16MHz: 3.3V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - ♦ Internal High Speed 8/12/16MHz RC – HIRC
  - ♦ External Low Speed 32.768kHz Crystal – LXT
  - ♦ Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 4K×16
- RAM Data Memory: 512×8
- True EEPROM Memory: 512×8
- Up to 16 touch key functions – fully integrated without requiring external components
- Watchdog Timer function
- Up to 26 bidirectional I/O lines
- Programmable I/O port source current and sink current for LED driver applications
- Single pin-shared external interrupt
- Multiple Timer Modules for time measurement, compare match output or PWM output or single pulse output function
- I^2C Interface
- Fully-duplex / Half-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- Software controlled up to 26-SCOM/SSEG lines LCD driver with 1/3 bias
- Dual Time Base functions for generation of fixed time interrupt signals
- Low voltage reset function
- Low voltage detect function
- Package type: 46-pin QFN

NFC Reader Features

- Highly integrated analog circuits for demodulation and decoding
- Operating voltage range: 2.3V~5.5V
- Ultra low power consumption, typical communication distance up to 10cm depending on antenna size
- Supports ISO/IEC 14443 TypeA/TypeB protocols
- Supports ISO 14443 A, B high transfer speed of 106kbit/s, 212kbit/s, 424kbit/s and 848kbit/s
- Supported host interfaces
  - I^{2}C interface up to 400kbit/s
  - Serial UART up to 1228.8kbit/s
- FIFO buffer for 64-byte transmission and reception
- Flexible interrupt modes
- Programmable timer
- 3 power saving modes: hardware power-down, software power-down and transmitter power-down
- Internal temperature sensor automatically stops RF transmission in high chip temperature situations
- Multiple independent power supplies to avoid mutual interference between blocks and improve the operation stability
- CRC coprocessor to implement CRC and parity function
- Internal oscillator for connection to a 27.12MHz quartz crystal
- Supports Low Power Card Detection (LPCD) function

General Description

The BS65F2042 is a Flash Memory type 8-bit high performance RISC architecture microcontroller with fully integrated touch key and NFC reader functions. With all touch key functions provided internally, completely eliminating the need for external components, the device has all the feactures to offer designers a reliable and easy means of implementing touch keys within their products applications.

For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

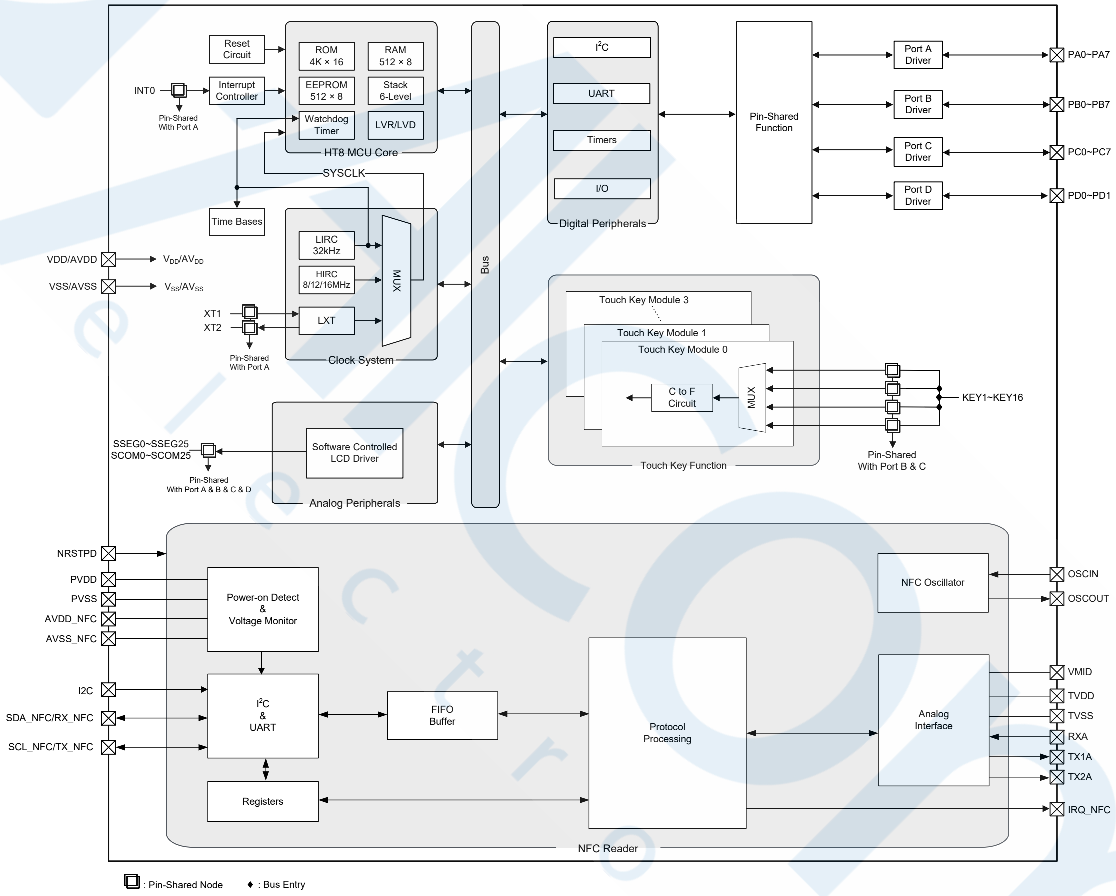
Analog features include a software controlled LCD function. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully intergrated UART or I^{2}C interface functions, two popular interfaces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

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

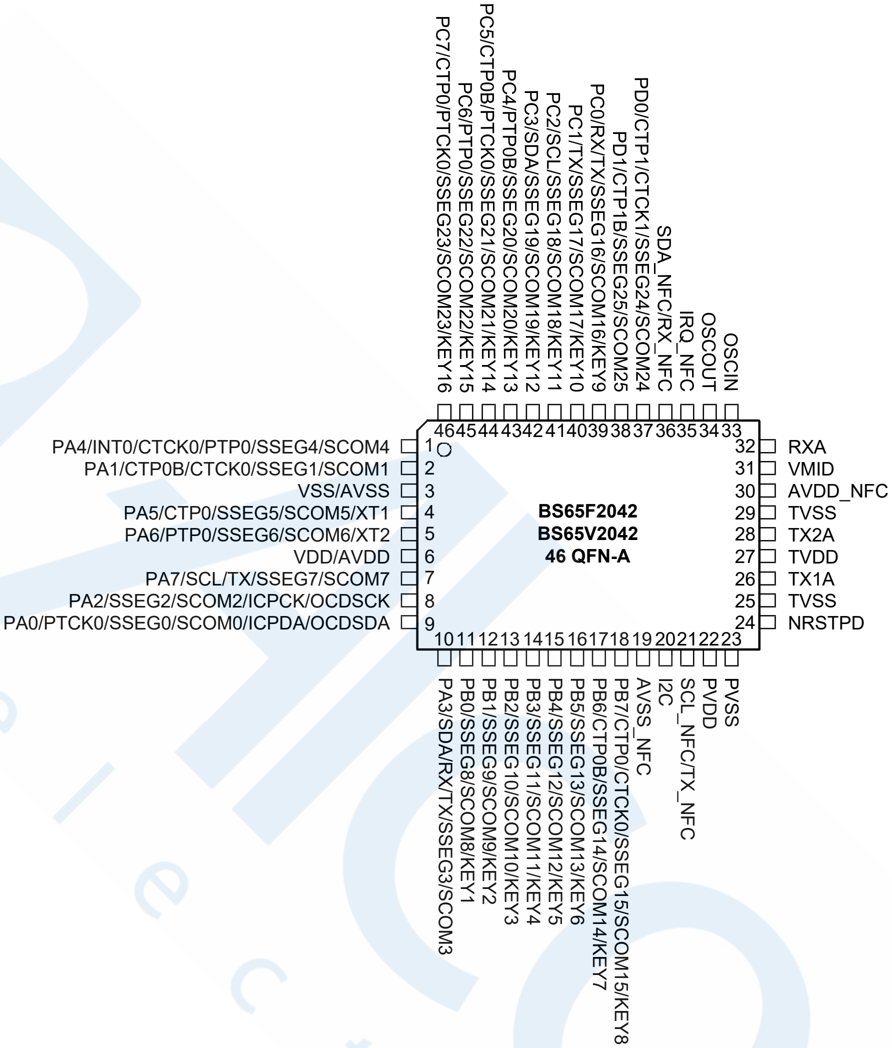
The device includes a 13.56MHz multi-Standard NFC reader. This NFC reader supports ISO/IEC 14443 A, ISO/IEC 14443B and Crypto_M modes. The internal transmitter is able to drive the reader antenna to communicate with ISO/IEC 14443 A, ISO/IEC1444B, Crypto_M cards and transponders without requiring additional active circuitry. The receiver block provides a robust and efficient implementation to demodulate and decode signals from ISO/IEC 14443 A, ISO/IEC1444B, Crypto_M cards and transponders. The digital block manages the complete ISO/IEC 14443 A framing detection and error detection including parity and CRC check.

The inclusion of flexible I/O programming features, Time Base funcitons along with many other features ensure that the device will find excellent use in applications such as smart door locks, hotel door locks, access control applications, smart home appliances, card readers and toys etc.

Block Diagram



Pin Assignment



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

2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the BS65V2042 device which is the OCDS EV chip for the BS65F2042 device.

Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/PTCK0/SSEG0/ SCOM0/ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTCK0	PAS0 IFS0	ST	—	PTM0 clock input
	SSEG0	PAS0	—	AN	Software controlled LCD segment output
	SCOM0	PAS0	—	AN	Software controlled LCD common output
	ICPDA	—	ST	CMOS	ICP Data/Address pin
	OCDSDA	—	ST	CMOS	OCDS Data/Address pin, for EV chip only
PA1/CTP0B/CTCK0/ SSEG1/SCOM1	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP0B	PAS0	—	CMOS	CTM0 inverted output
	CTCK0	PAS0 IFS0	ST	—	CTM0 clock input
	SSEG1	PAS0	—	AN	Software controlled LCD segment output
	SCOM1	PAS0	—	AN	Software controlled LCD common output
PA2/SSEG2/SCOM2/ ICPCK/OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SSEG2	PAS0	—	AN	Software controlled LCD segment output
	SCOM2	PAS0	—	AN	Software controlled LCD common output
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/SDA/RX/TX/ SSEG3/SCOM3	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SDA	PAS0 IFS0	ST	NMOS	I^{2}C data line
	RX/TX	PAS0 IFS0	ST	CMOS	UART serial data intput in full-duplex communication or UART serial data input/output in single wire mode communication
	SSEG3	PAS0	—	AN	Software controlled LCD segment output
	SCOM3	PAS0	—	AN	Software controlled LCD common output
PA4/INT0/CTCK0/PTP0/ SSEG4/SCOM4	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT0	PAS1	AN	—	External interrupt 0
	CTCK0	PAS1 IFS0	ST	—	CTM0 clock input
	PTP0	PAS1	—	CMOS	PTM0 output
	SSEG4	PAS1	—	AN	Software controlled LCD segment output
	SCOM4	PAS1	—	AN	Software controlled LCD common output

Pin Name	Function	OPT	I/T	O/T	Description
PA5/CTP0/SSEG5/SCOM5/XT1	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP0	PAS1	—	CMOS	CTM0 output
	SSEG5	PAS1	—	AN	Software controlled LCD segment output
	SCOM5	PAS1	—	AN	Software controlled LCD common output
	XT1	PAS1	AN	—	LXT oscillator pin
PA6/PTP0/SSEG6/SCOM6/XT2	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP0	PAS1	—	CMOS	PTM0 output
	SSEG6	PAS1	—	AN	Software controlled LCD segment output
	SCOM6	PAS1	—	AN	Software controlled LCD common output
	XT2	PAS1	—	AN	LXT oscillator pin
PA7/SCL/TX/SSEG7/SCOM7	PA7	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SCL	PAS1 IFS0	ST	NMOS	I^{2}C clock line
	TX	PAS1	—	CMOS	UART serial data output
	SSEG7	PAS1	—	AN	Software controlled LCD segment output
	SCOM7	PAS1	—	AN	Software controlled LCD common output
PB0/SSEG8/SCOM8/KEY1	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG8	PBS0	—	AN	Software controlled LCD segment output
	SCOM8	PBS0	—	AN	Software controlled LCD common output
	KEY1	PBS0	AN	—	Touch key input
PB1/SSEG9/SCOM9/KEY2	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG9	PBS0	—	AN	Software controlled LCD segment output
	SCOM9	PBS0	—	AN	Software controlled LCD common output
	KEY2	PBS0	AN	—	Touch key input
PB2/SSEG10/SCOM10/KEY3	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG10	PBS0	—	AN	Software controlled LCD segment output
	SCOM10	PBS0	—	AN	Software controlled LCD common output
	KEY3	PBS0	AN	—	Touch key input
PB3/SSEG11/SCOM11/KEY4	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG11	PBS0	—	AN	Software controlled LCD segment output
	SCOM11	PBS0	—	AN	Software controlled LCD common output
	KEY4	PBS0	AN	—	Touch key input
PB4/SSEG12/SCOM12/KEY5	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG12	PBS1	—	AN	Software controlled LCD segment output
	SCOM12	PBS1	—	AN	Software controlled LCD common output
	KEY5	PBS1	AN	—	Touch key input

Pin Name	Function	OPT	I/T	O/T	Description
PB5/SSEG13/SCOM13/KEY6	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG13	PBS1	—	AN	Software controlled LCD segment output
	SCOM13	PBS1	—	AN	Software controlled LCD common output
	KEY6	PBS1	AN	—	Touch key input
PB6/CTP0B/SSEG14/SCOM14/KEY7	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP0B	PBS1	—	CMOS	CTM0 inverted output
	SSEG14	PBS1	—	AN	Software controlled LCD segment output
	SCOM14	PBS1	—	AN	Software controlled LCD common output
	KEY7	PBS1	AN	—	Touch key input
PB7/CTP0/CTCK0/SSEG15/SCOM15/KEY8	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP0	PBS1	—	CMOS	CTM0 output
	CTCK0	PBS1 IFS0	ST	—	CTM0 clock input
	SSEG15	PBS1	—	AN	Software controlled LCD segment output
	SCOM15	PBS1	—	AN	Software controlled LCD common output
	KEY8	PBS1	AN	—	Touch key input
PC0/RX/TX/SSEG16/SCOM16/KEY9	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	RX/TX	PCS0 IFS0	ST	CMOS	UART serial data intput in full-duplex communication or UART serial data input/output in single wire mode communication
	SSEG16	PCS0	—	AN	Software controlled LCD segment output
	SCOM16	PCS0	—	AN	Software controlled LCD common output
	KEY9	PCS0	AN	—	Touch key input
PC1/TX/SSEG17/SCOM17/KEY10	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TX	PCS0	—	CMOS	UART serial data output
	SSEG17	PCS0	—	AN	Software controlled LCD segment output
	SCOM17	PCS0	—	AN	Software controlled LCD common output
	KEY10	PCS0	AN	—	Touch key input
PC2/SCL/SSEG18/SCOM18/KEY11	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SCL	PCS0 IFS0	ST	NMOS	I^2C clock line
	SSEG18	PCS0	—	AN	Software controlled LCD segment output
	SCOM18	PCS0	—	AN	Software controlled LCD common output
	KEY11	PCS0	AN	—	Touch key input
PC3/SDA/SSEG19/SCOM19/KEY12	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SDA	PCS0 IFS0	ST	NMOS	I^2C data line
	SSEG19	PCS0	—	AN	Software controlled LCD segment output
	SCOM19	PCS0	—	AN	Software controlled LCD common output
	KEY12	PCS0	AN	—	Touch key input

Pin Name	Function	OPT	I/T	O/T	Description
PC4/PTP0B/SSEG20/ SCOM20/KEY13	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP0B	PCS1	—	CMOS	PTM0 inverted output
	SSEG20	PCS1	—	AN	Software controlled LCD segment output
	SCOM20	PCS1	—	AN	Software controlled LCD common output
	KEY13	PCS1	AN	—	Touch key input
PC5/CTP0B/PTCK0/ SSEG21/SCOM21/KEY14	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP0B	PCS1	—	CMOS	CTM0 inverted output
	PTCK0	PCS1 IFS0	ST	—	PTM0 clock input
	SSEG21	PCS1	—	AN	Software controlled LCD segment output
	SCOM21	PCS1	—	AN	Software controlled LCD common output
	KEY14	PCS1	AN	—	Touch key input
PC6/PTP0/SSEG22/ SCOM22/KEY15	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP0	PCS1	—	CMOS	PTM0 output
	SSEG22	PCS1	—	AN	Software controlled LCD segment output
	SCOM22	PCS1	—	AN	Software controlled LCD common output
	KEY15	PCS1	AN	—	Touch key input
PC7/CTP0/PTCK0/ SSEG23/SCOM23/KEY16	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP0	PCS1	—	CMOS	CTM0 output
	PTCK0	PCS1 IFS0	ST	—	PTM0 clock input
	SSEG23	PCS1	—	AN	Software controlled LCD segment output
	SCOM23	PCS1	—	AN	Software controlled LCD common output
	KEY16	PCS1	AN	—	Touch key input
PD0/CTP1/CTCK1/ SSEG24/SCOM24	PD0	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP1	PDS0	—	CMOS	CTM1 output
	CTCK1	PDS0	ST	—	CTM1 clock input
	SSEG24	PDS0	—	AN	Software controlled LCD segment output
	SCOM24	PDS0	—	AN	Software controlled LCD common output
PD1/CTP1B/SSEG25/ SCOM25	PD1	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP1B	PDS0	—	CMOS	CTM1 inverted output
	SSEG25	PDS0	—	AN	Software controlled LCD segment output
	SCOM25	PDS0	—	AN	Software controlled LCD common output
I2C	I2C	—	ST	—	NFC I^{2}C interface enable
SCL_NFC/TX_NFC	SCL_NFC	—	ST	NMOS	NFC I^{2}C clock line
	TX_NFC	—	—	CMOS	NFC UART transmitter pin
SDA_NFC/RX_NFC	SDA_NFC	—	ST	NMOS	NFC I^{2}C data line
	RX_NFC		ST	—	NFC UART receiver pin
NRSTPD	NRSTPD	—	ST	—	NFC reset or power-down pin, low active
TX1A	TX1A	—	—	CMOS	NFC transmitter antenna pin 1
TX2A	TX2A	—	—	CMOS	NFC transmitter antenna pin 2
VMID	VMID	—	PWR	—	NFC internal reference voltage
RXA	RXA	—	ST	—	NFC receiver antenna pin
OSCIN	OSCIN	—	ST	—	NFC 27.12MHz crystal input

Pin Name	Function	OPT	I/T	O/T	Description
OSCOUT	OSCOUT	—	—	CMOS	NFC 27.12MHz crystal output
IRQ_NFC	IRQ_NFC	—	—	CMOS	NFC interrupt request output
PVDD	PVDD	—	PWR	—	NFC pin and digital positive power supply
PVSS	PVSS	—	PWR	—	NFC pin and digital negative power supply, ground
TVDD	TVDD	—	PWR	—	NFC transmitter positive power supply
TVSS	TVSS	—	PWR	—	NFC transmitter negative power supply, ground
AVDD_NFC	AVDD_NFC	—	PWR	—	NFC analog positive power supply
AVSS_NFC	AVSS_NFC	—	PWR	—	NFC analog negative power supply, ground
VDD/AVDD	VDD	—	PWR	—	Positive power supply
	AVDD	—	PWR	—	Analog positive power supply
VSS/AVSS	VSS	—	PWR	—	Negative power supply, ground
	AVSS	—	PWR	—	Analog negative power supply, ground

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

Absolute Maximum Ratings

MCU Supply Voltage	V_{SS}−0.3V to 6.0V
MCU Input Voltage	V_{SS}−0.3V to V_{DD}+0.3V
NFC Analog Power Supply	V_{SS}−0.5V to 6.0V
NFC PVDD Power Supply	V_{SS}−0.5V to 6.0V
NFC TVDD Power Supply	V_{SS}−0.5V to 6.0V
Storage Temperature	-60°C to 150°C
Operating Temperature	-40°C to 85°C
I_{OH} Total	-80mA
I_{OL} Total	100mA
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 the 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~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	f_{SYS}=8MHz	1.8	—	5.5	V
		f_{SYS}=12MHz	2.7	—	5.5	
		f_{SYS}=16MHz	3.3	—	5.5	
	Operating Voltage – LXT	f_{SYS}=32768Hz	1.8	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32kHz	1.8	—	5.5	V

Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	1.8V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3	5	6	
	IDLE0 Mode – LIRC	1.8V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE0 Mode – LXT	1.8V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE1 Mode – HIRC	1.8V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	540	750	900	
		5V		—	800	1200	1400	
		3.3V	f_{SUB} on, f_{SYS}=16MHz	—	0.80	1.20	1.44	mA
		5V		—	1.4	2.0	2.4	

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

1. Any digital inputs are set 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=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	1.8V	f_{SYS}=32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	SLOW Mode – LXT	1.8V	f_{SYS}=32768Hz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	1.8V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	
		2.7V	f_{SYS}=12MHz	—	1.0	1.4	
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	
		3.3V	f_{SYS}=16MHz	—	1.5	3.0	
		5V		—	2.5	5.0	

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

1. Any digital inputs are set 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.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C ~ 85°C	-2%	8	+2%	
		1.8V~ 5.5V	-40°C ~ 85°C	-10%	8	+10%	
		2.2V~ 5.5V	25°C	-3.5%	8	+3.5%	
			-40°C ~ 85°C	-5%	8	+5%	
		2.7V~ 5.5V	25°C	-2.5%	8	+2.5%	
			-40°C ~ 85°C	-3%	8	+3%	
	12MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	12	+1%	MHz
			-40°C ~ 85°C	-2%	12	+2%	
		2.7V~ 5.5V	25°C	-2.5%	12	+2.5%	
			-40°C ~ 85°C	-3%	12	+3%	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	16MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	16	+1%	MHz
			-40°C~85°C	-2%	16	+2%	
		3.3V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C~ 85°C	-3%	16	+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 1.8V 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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	Oscillator Frequency	2.2V~5.5V	25°C	-10%	32	+10%	kHz
			-40°C ~ 85°C	-50%	32	+60%	
		1.8V~5.5V	25°C	-50%	32	+10%	
t_{START}	LIRC Start Up Time	—	—	—	—	500	μs

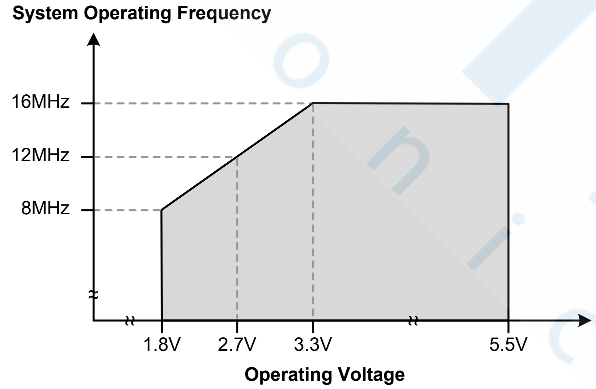
External Low Speed Crystal Oscillator Characteristics – LXT

Ta=25°C

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	—	—	40	—	60	%
t_{START}	LXT Start Up Time	3V	—	—	—	1000	ms
		5V	—	—	—	1000	ms
R_{NEG}	Negative Resistance (Note)	1.8V	—	3×ESR	—	—	Ω

Note: C1, C2 and R_P are external components. C1=C2=10pF. R_P=10MΩ. C_L=7pF, ESR=30kΩ.

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time (Wake-up from Condition where f_{SYS} is off)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LXT}$	—	1024	—	t_{LXT}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time (Wake-up from Condition where f_{SYS} is on)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LXT}$ or f_{LIRC}	—	2	—	t_{SUB}
	System Speed Switch Time (FAST to SLOW Mode or SLOW to FAST Mode)	—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
		—	f_{LXT} switches from off → on	—	1024	—	t_{LXT}
t_{RSTD}	System Reset Delay Time (Reset source from Power-on Reset or LVR Hardware Reset)	—	RR_{POR}=5V/ms	10	16	24	ms
	System Reset Delay Time (LVRC/WDTC/RSTC Software Reset)	—	—				
	System Reset Delay Time (Reset source from WDT Overflow)	—	—				
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	180	μs

Note: 1. For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.

2. The time units, shown by the symbols, t_{HIRC}, etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example $t_{HIRC}=1/f_{HIRC}$, $t_{SYS}=1/f_{SYS}$ etc.

3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.

4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Ports Except Port PB	3V	$V_{OL}=0.1V_{DD}$, PxNS=0	16	32	—	mA
			$V_{OL}=0.1V_{DD}$, PxNS=1	25	50	—	
		5V	$V_{OL}=0.1V_{DD}$, PxNS=0	32	65	—	
			$V_{OL}=0.1V_{DD}$, PxNS=1	50	100	—	
	Sink Current for PB Port	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH}	Source Current for I/O Ports	3V	V_{OH}=0.9V_{DD}, SLEDCn[m+1:m]=00B, (n=0, 1; m=0, 2, 4 or 6)	-0.7	-1.5	—	mA
		5V		-1.5	-2.9	—	
		3V	V_{OH}=0.9V_{DD}, SLEDCn[m+1:m]=01B, (n=0, 1; m=0, 2, 4 or 6)	-1.3	-2.5	—	
		5V		-2.5	-5.1	—	
		3V	V_{OH}=0.9V_{DD}, SLEDCn[m+1:m]=10B, (n=0, 1; m=0, 2, 4 or 6)	-1.8	-3.6	—	
		5V		-3.6	-7.3	—	
		3V	V_{OH}=0.9V_{DD}, SLEDCn[m+1:m]=11B, (n=0, 1; m=0, 2, 4 or 6)	-4	-8	—	
		5V		-8	-16	—	
R_{PH}	Pull-high Resistance for I/O Ports (Note)	3V	LVPU=0, PxPU=FFH (Px: PA, PB, PC, PD)	20	60	100	kΩ
		5V		10	30	50	
		3V	LVPU=1, PxPU=FFH (Px:PA, PB, PC, PD)	6.67	15	23	
		5V		3.5	7.5	12.0	
I_{LEAK}	Input Leakage Current	5V	V_{IN}=V_{DD} or V_{IN}=V_{SS}	—	—	±1	μA
t_{INT}	Interrupt Input Pin Minimum Pulse Width	—	—	10	—	—	μs
t_{TCK}	TM Clock Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs

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

Memory Characteristics

Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Flash Program Memory							
V_{DD}	V_{DD} for Read and Write	—	—	1.8	—	5.5	V
t_{FER}	Erase Time	—	—	2.273	2.500	2.778	ms
t_{FWR}	Write Time	—	—	1.364	1.500	1.667	ms
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
t_{ACTV}	Activation Time – Wake-up from IDLE/SLEEP Mode	—	—	32	—	64	μs
Data EEPROM Memory							
V_{DD}	V_{DD} for Read and Write	—	—	1.8	—	5.5	V
t_{EERD}	EEPROM Read Time	—	—	—	—	4	t_{SYS}
t_{EEWR}	Write Time (Byte Mode)	—	EWERTS=0	—	5.4	8.6	ms
		—	EWERTS=1	—	6.7	10.6	ms
	Write Time (Page Mode)	—	EWERTS=0	—	2.2	3.6	ms
		—	EWERTS=1	—	3.0	4.8	ms
t_{EEER}	EEPROM Erase Time	—	EWERTS=0	—	3.2	5.2	ms
		—	EWERTS=1	—	3.7	6.0	ms
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	Data Retention Time	—	Ta=25°C	—	40	—	Year

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

Note: 1. "E/W" means Erase/Write times.

2. The ROM activation time t_{ACTV} should be added when calculating the total system start-up time of a wake-up from the IDLE/SLEEP mode.

LVR/LVD Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	1.8	—	5.5	V
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 1.7V	-5%	1.7	+5%	V
		—	LVR enable, voltage select 1.9V	-5%	1.9	+5%	
		—	LVR enable, voltage select 2.55V	-3%	2.55	+3%	
		—	LVR enable, voltage select 3.15V	-3%	3.15	+3%	
		—	LVR enable, voltage select 3.8V	-3%	3.8	+3%	
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 1.8V	-5%	1.8	+5%	V
		—	LVD enable, voltage select 2.0V		2.0		
		—	LVD enable, voltage select 2.4V		2.4		
		—	LVD enable, voltage select 2.7V		2.7		
		—	LVD enable, voltage select 3.0V		3.0		
		—	LVD enable, voltage select 3.3V		3.3		
		—	LVD enable, voltage select 3.6V		3.6		
		—	LVD enable, voltage select 4.0V		4.0		
I_{LVRLVD}	Operating Current	3V	LVD enable, LVR enable, V_{LVR}=1.9V, V_{LVD}=2V	—	—	10	μA
		5V		—	10	15	
t_{LVDS}	LVDO Stable Time	—	LVR enable, VBGEN=0, LVD off → on	—	—	18	μs
		—	LVR disable, VBGEN=0, 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	5V	LVD disable	—	—	14	μA
I_{LVD}	Additional Current for LVD Enable	5V	LVR disable	—	—	14	μA

Software Controlled LCD Driver Electrical Characteristics

Ta=-40°C~85°C

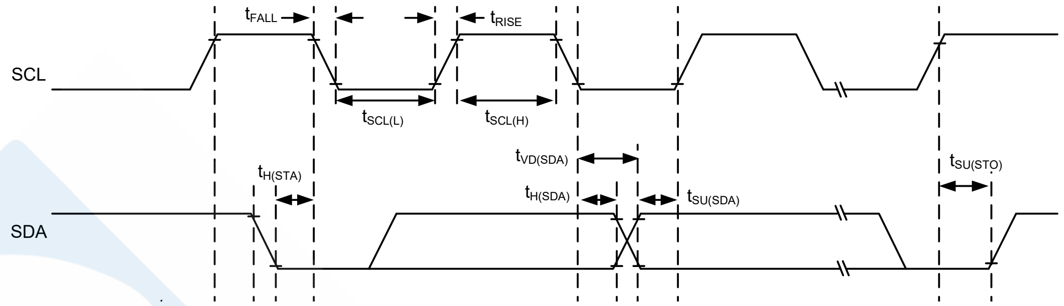
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{BIAS}	LCD Bias Current	5V	ISEL[1:0]=00B	5.81	8.30	10.79	μA
			ISEL[1:0]=01B	11.62	16.60	21.58	
			ISEL[1:0]=10B	35	50	65	
			ISEL[1:0]=11B	70	100	130	
V_{SCOM}	V_{DD}×2/3 Voltage for LCD SCOM Output	2.2V~ 5.5V	No load	$0.305V_{DD}$	$0.33V_{DD}$	$0.355V_{DD}$	V
	V_{DD}×1/3 Voltage for LCD SCOM Output	2.2V~ 5.5V	No load	$0.31V_{DD}$	$0.33V_{DD}$	$0.35V_{DD}$	V

I^2C Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{I2C}	I^2C Standard Mode (100kHz) f_{SYS} Frequency (Note)	—	No clock debounce	2	—	—	MHz
			2 system clock debounce	4	—	—	
			4 system clock debounce	4	—	—	
	I^2C Fast Mode (400kHz) f_{SYS} Frequency (Note)	—	No clock debounce	4	—	—	MHz
			2 system clock debounce	8	—	—	
			4 system clock debounce	8	—	—	
f_{SCL}	SCL Clock Frequency	3V/5V	Standard mode	—	—	100	kHz
			Fast mode	—	—	400	
$t_{SCL(H)}$	SCL Clock High Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
$t_{SCL(L)}$	SCL Clock Low Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
t_{FALL}	SCL and SDA Fall Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
t_{RISE}	SCL and SDA Rise Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
$t_{SU(SDA)}$	SDA Data Setup Time	3V/5V	Standard mode	0.25	—	—	μs
			Fast mode	0.1	—	—	
$t_{H(SDA)}$	SDA Data Hold Time	3V/5V	—	0.1	—	—	μs
$t_{VD(SDA)}$	SDA Data Valid Time	3V/5V	—	—	—	0.6	μs
$t_{SU(STA)}$	Start Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	
$t_{H(STA)}$	Start Condition Hold Time	3V/5V	Standard mode	4.0	—	—	μs
			Fast mode	0.6	—	—	
$t_{SU(STO)}$	Stop Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	

Note: Using the debounce function can make the transmission more stable and reduce the probability of communication failure due to interference.



I²C Timing Diagram

NFC Reader Electrical Characteristices

NFC Operating Condition

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
AV_{DD}	Analog Power Supply	$PV_{SS}=AV_{SS}=TV_{SS}=0V$, $PV_{DD}\leq AV_{DD}\leq TV_{DD}$	2.3	3.3	5.5	V
TV_{DD}	Transmitter Power Supply		2.3	3.3	5.5	V
PV_{DD}	Pin and Digital Power Supply		2.0	3.3	5.5	V
Ta	Operating Temperature	—	-40	—	85	°C

Note: Recommended power supply condition: $PV_{DD}\leq AV_{DD}\leq TV_{DD}$. The performance of other power supply conditions are not guaranteed.

NFC Power Consumption Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
3.3V Characteristics						
I_{HPD}	Hardware Power-down Current	$AV_{DD}=TV_{DD}=PV_{DD}=3.3V$, NRSTPD=LOW	—	0.02	1.00	μA
I_{SPD}	Software Power-down Current	$AV_{DD}=TV_{DD}=PV_{DD}=3.3V$, RF level detector on	—	0.6	5.0	μA
I_{IDLE}	Idle Current	$AV_{DD}=TV_{DD}=PV_{DD}=3.3V$	—	3.52	5.00	mA
I_{RXA}	Receving Current	$AV_{DD}=TV_{DD}=PV_{DD}=3.3V$	—	6.5	10.0	mA
I_{PVDD}	Pin and Digital Supply Current	$PV_{DD}=3.3V$	—	2.97	4.00	mA
I_{AVDD}	Analog Supply Current	$AV_{DD}=3.3V$, RcvOff=0	—	2.98	6.00	mA
		$AV_{DD}=3.3V$, RcvOff=1	—	2.95	6.00	mA
I_{TVDD}	Transmitter Supply Current	Continuous transmit carrier, $TV_{DD}=3.3V$	—	60	160	mA
I_{LPCD}	Low Power Card Detection Current	$AV_{DD}=TV_{DD}=PV_{DD}=3.3V$, Average current consumption in LPCD mode @ WUPeriod=500ms & detect wave time=5.9μs	—	1.5	10.0	μA
V_{Ripple}	Power Supply Ripple Rejection	—	—	—	400	mV
V_{Noise}	Power Supply Random Noise Rejection	—	—	—	1600	mV
t_{OSU}	Oscillator Start Up Time	—	—	300	—	μs
5V Characteristics						
I_{HPD}	Hardware Power-down Current	$AV_{DD}=TV_{DD}=PV_{DD}=5V$, NRSTPD=LOW	—	0.02	1.00	μA
I_{SPD}	Software Power-down Current	$AV_{DD}=TV_{DD}=PV_{DD}=5V$, RF level detector on	—	0.8	5.0	μA
I_{IDLE}	Idle Current	$AV_{DD}=TV_{DD}=PV_{DD}=5V$	—	3.52	5.00	mA

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
I_{RXA}	Receving Current	AV_{DD}=TV_{DD}=PV_{DD}=5V	—	6.55	10.00	mA
I_{PVDD}	Pin and Digital Supply Current	PV_{DD}=5V	—	3.2	5.0	mA
I_{AVDD}	Analog Supply Current	AV_{DD}=5V, RcvOff=0	—	3.1	6.0	mA
		AV_{DD}=5V, RcvOff=1	—	3.07	6.00	mA
I_{TVDD}	Transmitter Supply Current	Continuous transmit carrier, TV_{DD}=5V	—	90	230	mA
V_{Ripple}	Power Supply Ripple Rejection	—	—	—	300	mV
V_{Noise}	Power Supply Random Noise Rejection	—	—	—	1600	mV
t_{OSU}	Oscillator Start Up Time	—	—	300	—	μs

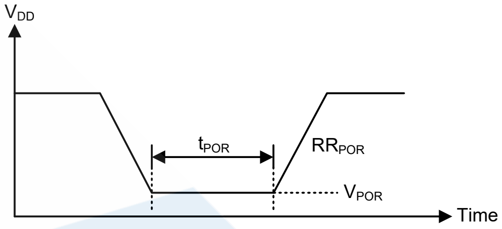
NFC Input/Output Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
Pins I2C, NRSTPD, SDA_NFC/RX_NFC						
I_{LEAK}	Input Leakage Current	—	-1	—	1	μA
V_{IH}	High Level Input Voltage	—	0.7PV_{DD}	—	—	V
V_{IL}	Low Level Input Voltage	—	—	—	0.3PV_{DD}	V
Pins SCL_NFC/TX_NFC, IRQ_NFC						
I_{LEAK}	Input Leakage Current	Connected to VDD when in operation/idle state	-1	—	1	μA
		Connected to GND when in operation/idle state	—	13.5	—	μA
		Connected to 1.5V when in operation/idle state	—	13	—	μA
		Left floating when in operation/idle state	—	0	—	μA
V_{IH}	High Level Input Voltage	—	0.7PV_{DD}	—	—	V
V_{IL}	Low Level Input Voltage	—	—	—	0.3PV_{DD}	V
V_{OH}	High Level Output Voltage	—	PV_{DD}-0.4	—	PV_{DD}	V
V_{OL}	Low Level Output Voltage	—	PV_{SS}	—	PV_{SS}+0.4	V
I_{OH}	High Level Output Current	PV_{DD}=3V	—	—	5	mA
I_{OL}	Low Level Output Current	PV_{DD}=3V	—	—	14	mA
Pins OSCIN, OSCOUT Connection Requirements						
f_{xtal}	Crystal Frequency	—	—	27.12	—	MHz
	Frequency Tolerance	—	—	±10	±20	ppm
ESR	Equivalent Series Resistance	—	—	—	100	Ω
C_L	Load Capacitance	—	—	10	—	pF
P_{xtal}	Crystal Power Dissipation	—	—	50	100	mW

Power-on Reset Characteristics

Ta=-40°C~85°C

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



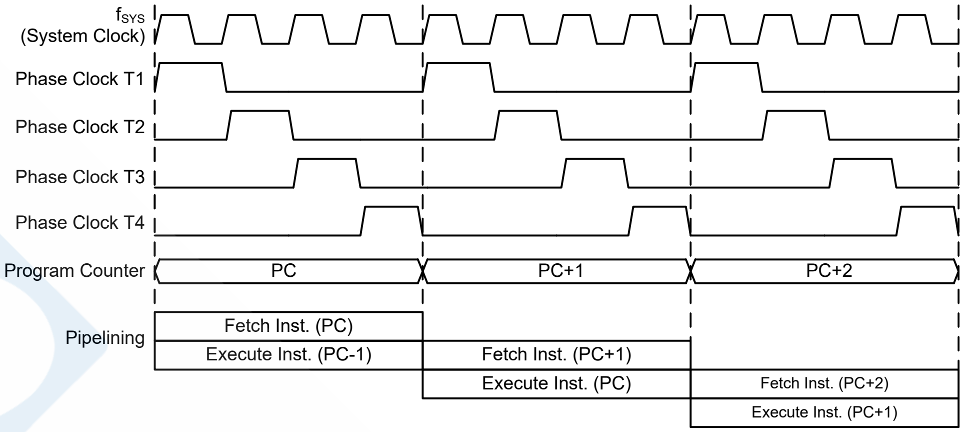
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The 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, with the exception of branch or call instructions which need one more cycle. An 8-bit wide ALU is used in practically all instruction set operations, which carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

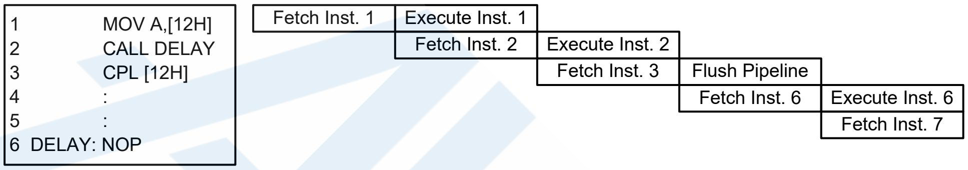
Clocking and Pipelining

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

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



System Clocking and Pipelining



Instruction Fetching

Program Counter

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

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

Program Counter	
Program Counter High Byte	**PCL Register**
PC11~PC8	PCL7~PCL0

Program Counter

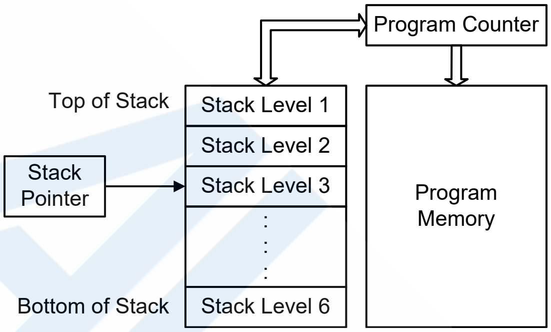
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 6 levels and 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,

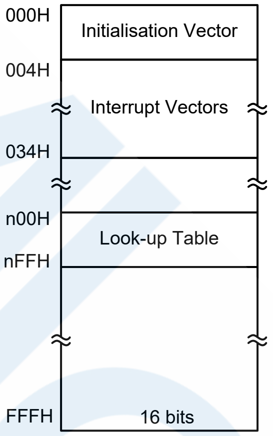
  LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

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 offers 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 4K×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 set 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 set by placing the address of the look up data to be retrieved in the table pointer register, TBLP and TBHP. These registers define the total address of the look-up table.

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

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

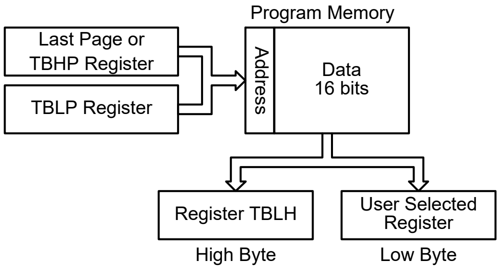


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "0F00H" which refers to the start address of the last page within the 4K words Program Memory. The table pointer low byte register is set 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 "0F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by the TBLP and TBHP registers if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

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

Table Read Program Example

```
ds .section 'data'
tempreg1 db?          ; temporary register #1
tempreg2 db?          ; temporary register #2
code0 .section 'code'
mov a,06h             ; initialise table pointer - note that this address is referenced
mov tblp,a            ; to the last page or the page that tbhp pointed
mov a,0FH             ; initialise high table pointer
mov tbhp,a            ; it is not necessary to set tbhp if executing tabrdl or ltabrdl
:
:
tabrd tempreg1        ; transfers value in table referenced by table pointer
                      ; data at program memory address "0F06H" transferred to tempreg1
dec tblp              ; and TBLH reduce value of table pointer by one
tabrd tempreg2        ; transfers value in table referenced by table pointer
                      ; data at program memory address "0F05H" transferred to tempreg2
                      ; and TBLH in this example the data "1AH" is transferred to
                      ; tempreg1 and data "0FH" to tempreg2
                      ; the value "00H" will be transferred to the high byte register TBLH
:
:
code1 .section 'code'
org 0F00H             ; sets initial address of last page
dc  00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

In Circuit Programming – ICP

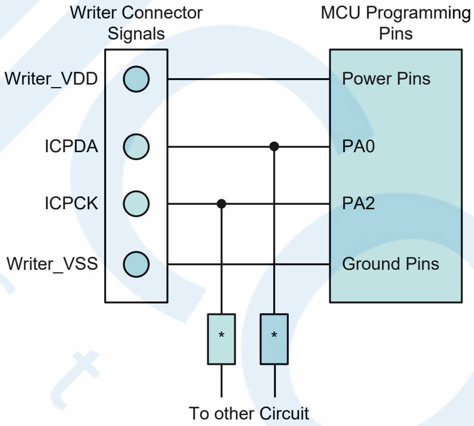
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VDD	VDD, AVDD_NFC&TVDD&PVDD&NRSTPD	Power Supply
VSS	VSS, AVSS_NFC&TVSS&&PVSS	Ground

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

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



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

On-Chip Debug Support – OCDS

There is EV chip named BS65V2042 which is used to emulate the real MCU device named BS65F2042. The EV chip device also provides an “On-Chip Debug” function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for “On-Chip Debug” function. Users can use the OCDS function to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip device for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named “Holtek e-Link for 8-bit MCU OCDS User’s Guide”.

Holtek e-Link Pins	EV Chip Pins	Pin Description
OCDSDA	OCDSDA	On-Chip Debug Support Data/ Address input/output
OCDSCK	OCDSCK	On-Chip Debug Support Clock input
VDD	VDD, AVDD_NFC&TVDD&PVDD&NRSTPD	Power Supply
VSS	VSS, AVSS_NFC&TVSS&&PVSS	Ground

Data Memory

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

Categorized into two types, the first of these is an area of RAM where the 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.

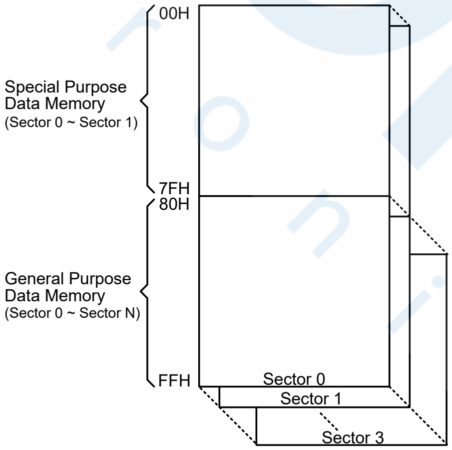
Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value when using the indirectly accessing method.

Structure

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

Special Purpose Data Memory	General Purpose Data Memory	
Located Sectors	Capacity	Sector: Address
0, 1	512×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH

Data Memory Summary



Data Memory Structure

Data Memory Addressing

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

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

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0	Sector 1		Sector 0	Sector 1
00H	IAR0	PAS0	40H	PDPU	EEC
01H	MP0	PAS1	41H	EEAL	
02H	IAR1	PBS0	42H	EEAH	
03H	MP1L	PBS1	43H	EED	
04H	MP1H	PCS0	44H	TKTMR	
05H	ACC	PCS1	45H	TKC0	
06H	PCL	PDS0	46H	TK16DL	
07H	TBLP		47H	TK16DH	
08H	TBLH		48H	TKC1	
09H	TBHP		49H	TKM016DL	
0AH	STATUS		4AH	TKM016DH	
0BH			4BH	TKM0ROL	
0CH	IAR2		4CH	TKM0ROH	
0DH	MP2L		4DH	TKM0C0	
0EH	MP2H		4EH	TKM0C1	
0FH	RSTFC		4FH	TKM116DL	
10H	INTC0	PANS	50H	TKM116DH	
11H	INTC1	PCNS	51H	TKM1ROL	
12H	INTC2	PDNS	52H	TKM1ROH	
13H	INTC3		53H	TKM1C0	
14H	PA		54H	TKM1C1	
15H	PAC		55H	TKM216DL	
16H	PAPU		56H	TKM216DH	
17H	PAWU		57H	TKM2ROL	
18H	SLEDC0	PTM0C0	58H	TKM2ROH	
19H	SLEDC1	PTM0C1	59H	TKM2C0	
1AH		PTM0DL	5AH	TKM2C1	
1BH	TB0C	PTM0DH	5BH	TKM316DL	
1CH	PSC0R	PTM0AL	5CH	TKM316DH	
1DH	LVRC	PTM0AH	5DH	TKM3ROL	
1EH	LVPUC	PTM0RPL	5EH	TKM3ROH	
1FH	LXTC	PTM0RPH	5FH	TKM3C0	
20H	PB		60H	TKM3C1	
21H	PBC		61H		
22H	PBPU		62H		
23H	WDTC		63H		
24H	IICC0		64H		
25H	IICC1		65H		
26H	IICD		66H		
27H	IICA		67H		
28H	IICTOC		68H	ORMC	
29H	RSTC		69H	CTM0C0	
2AH	USR	SLCDC0	6AH	CTM0C1	
2BH	UCR1	SLCDS0	6BH	CTM0DL	
2CH	UCR2	SLCDS1	6CH	CTM0DH	
2DH	UCR3	SLCDS2	6DH	CTM0AL	
2EH	BRDH	SLCDS3	6EH	CTM0AH	
2FH	BRDL		6FH	CTM1C0	
30H	UFCR		70H	CTM1C1	
31H	TXR_RXR		71H	CTM1DL	
32H	RxCNT		72H	CTM1DH	
33H	IFS0		73H	CTM1AL	
34H			74H	CTM1AH	
35H			75H		
36H	SCC		76H		
37H	LVDC		77H		
38H	HIRCC		78H		
39H	PC		79H		
3AH	PCC		7AH		
3BH	PCPU		7BH		
3CH	MFI0		7CH		
3DH			7DH	INTEG	
3EH	PD		7EH	PSC1R	
3FH	PDC		7FH	TB1C	

▨ : 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 section; however several registers require a separate description in this section.

Indirect Addressing Registers – IAR0, IAR1, IAR2

The Indirect Addressing Registers, IAR0, IAR1 and IAR2, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0, IAR1 and IAR2 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0, MP1L/MP1H or MP2L/MP2H. Acting as a pair, IAR0 and MP0 can together access data only from Sector 0 while the IAR1 register together with 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 extended instructions 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              ; set size of block
    mov block, a
    mov a, offset adres1    ; Accumulator loaded with first RAM address
    mov mp0, a              ; set 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              ; set size of block
    mov  block, a
    mov  a, 01h              ; set the memory sector
    mov  mp1h, a
    mov  a, offset adres1    ; Accumulator loaded with first RAM address
    mov  mp1l, a             ; set 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 examples shown above, no reference is made to specific Data Memory addresses.

Direct Addressing Program Example using extended instructions

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

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

Accumulator – ACC

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

Program Counter Low Byte 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 set 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.

Option Memory Mapping Register – ORMC

The ORMC register is used to enable Option Memory Mapping function. The Option Memory capacity is 64 words. When a specific pattern of 55H and AAH is consecutively written into this register, the Option Memory Mapping function will be enabled and then the Option Memory code can be read by using the table read instruction. The Option Memory addresses 00H~3FH will be mapped to Program Memory last page addresses C0H~FFH.

To successfully enable the Option Memory Mapping function, the specific pattern of 55H and AAH must be written into the ORMC register in two consecutive instruction cycles. It is therefore recommended that the global interrupt bit EMI should first be cleared before writing the specific pattern, and then set high again at a proper time according to users’ requirements after the pattern is successfully written. An internal timer will be activated when the pattern is successfully written. The mapping operation will be automatically finished after a period of 4×t_{LIRC}. Therefore, users should read the data in time, otherwise the Option Memory Mapping function needs to be restarted. After the completion of each consecutive write operation to the ORMC register, the timer will recount.

When the table read instructions are used to read the Option Memory code, both “TABRD [m]” and “TABRDL [m]” instructions can be used. However, care must be taken if the “TABRD [m]” instruction is used, the table pointer defined by the TBHP register must be referenced to the last page. Refer to corresponding sections about the table read instruction for more details.

- **ORMC Register**

Bit	7	6	5	4	3	2	1	0
Name	ORMC7	ORMC6	ORMC5	ORMC4	ORMC3	ORMC2	ORMC1	ORMC0
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 **ORMC7~ORMC0**: Option Memory Mapping specific pattern
When a specific pattern of 55H and AAH is written into this register, the Option Memory Mapping function will be enabled. Note that the register content will be cleared after the MCU is woken up from the IDLE/SLEEP mode.

Status Register – STATUS

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

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

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

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

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

- **STATUS Register**

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

"x": unknown

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

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

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

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

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

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

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

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

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

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

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

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

EEPROM Data Memory

The 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 512×8 bits. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address register pair and a data register in Sector 0 and a single control register in Sector 1.

EEPROM Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEAL	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH	—	—	—	—	—	—	—	EEAH0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD

EEPROM Register List

- **EEAL Register**

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

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

- **EEAH Register**

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

Bit 7~1 Unimplemented, read as “0”

Bit 0 **EEAH0**: Data EEPROM high byte address bit 0

- **EED Register**

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

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

- **EEC Register**

Bit	7	6	5	4	3	2	1	0
Name	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD
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 **EWERTS**: Data EEPROM Erase time and Write time select
0: Erase time is 3.2ms (t_{EEER}) / Write time is 2.2ms (t_{EEWR})
1: Erase time is 3.7ms (t_{EEER}) / Write time is 3.0ms (t_{EEWR})

Bit 6 **EREN**: Data EEPROM Erase Enable
0: Disable
1: Enable

This bit is used to enable Data EEPROM erase function and must be set high before Data EEPROM erase operations are carried out. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Clearing this bit to zero will inhibit data EEPROM erase operations.

Bit 5 **ER**: Data EEPROM Erase Control
0: Erase cycle has finished
1: Activate an erase cycle

This is the Data EEPROM Erase Control Bit. When this bit is set high by the application program, an erase cycle will be activated. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Setting this bit high will have no effect if the EREN has not first been set high.

Bit 4 **MODE**: Data EEPROM operation mode selection
0: Byte operation mode
1: Page operation mode

This is the EEPROM operation mode selection bit. When the bit is set high by the application program, the Page write, erase or read function will be selected. Otherwise, the byte write or read function will be selected. The EEPROM page buffer size is 16-byte.

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. Note that this bit will automatically be clear to zero by the hardware after the write operation has finished.

Bit 2 **WR**: Data EEPROM Write Control
0: Write cycle has finished
1: Activate a write cycle

This is the Data EEPROM Write Control Bit. When this bit is set high by the application program, a write cycle will be actived. 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**: Data EEPROM Read Control
0: Read cycle has finished
1: Activate a read cycle

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

Note: 1. The EREN, ER, WREN, WR, RDEN and RD cannot be set high at the same time in one instruction.
2. Ensure that the f_{SUB} clock is stable before executing the erase or write operation.
3. Ensure that the erase or write operation is totally complete before changing the contents of the EEPROM related registers.

Read Operation from the EEPROM

Reading data from the EEPROM can be implemented by two modes for this device, byte read mode or page read mode, which is controlled by the EEPROM operation mode selection bit, MODE, in the EEC register.

Byte Read Mode

The EEPROM byte read operation can be executed if the mode selection bit, MODE, is cleared to 0. For a byte read operation the desired EEPROM address should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM byte read operation. Note that setting the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the read cycle terminates, the EEPROM data can be read from the EED register and the RD bit will automatically be cleared to zero. 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.

Page Read Mode

The page read operation can be executed if the mode selection bit, MODE, is set to 1. The page size can be up to 16 bytes for the page read operation. For a page read operation the desired start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM page read operation. Note that setting the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the current byte read cycle terminates, the EEPROM data can be read from the EED register and then the current address will be incremented by one by hardware. After this the RD bit will automatically be cleared to zero. The data which is stored in the next EEPROM address can continuously be read from the EED register when the RD bit is again set high without reconfiguring the EEPROM address and RDEN control bit. The application program can poll the RD bit to determine when the data is valid for reading.

The EEPROM address higher 5 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page read operation mode the lower 4-bit address value will automatically be incremented by one. However, the higher 5-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”.

Page Erase Operation to the EEPROM

The page erase operation can be executed if the mode selection bit, MODE, is set to 1.The EEPROM is capable of a 16-byte page erase. The internal page buffer will be cleared by hardware after power-on reset. When the EEPROM erase enable control bit, namely EREN, is changed from "1" to "0", the internal page buffer will also be cleared. Note that when the EREN bit is changed from "0" to "1", the internal page buffer will not be cleared. The EEPROM address higher 5 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page erase operation mode the lower 4-bit address value will automatically be incremented by one after each dummy data byte is written into the EED register. However, the higher 5-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, namely 0FH, the EEPROM address low 4-bit value will stop at 0FH. The EEPROM address will not "roll over".

For page erase operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers and the dummy data to be written is placed in the EED register. The maximum data length is 16-byte. Note that the write operation to the EED register is used to tag address, it must be implemented to determine which addresses to be erased. When the page dummy data is completely written, then the EREN bit in the EEC register should be set high to enable erase operations and the ER bit must be immediately set high to initiate the EEPROM erase process. These two instructions must be executed in two consecutive instruction cycles to activate an erase operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing an erase operation and then set again after a valid erase activation procedure has completed.

As the EEPROM erase 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 erased from the EEPROM. Detecting when the erase cycle has finished can be implemented either by polling the ER bit in the EEC register or by using the EEPROM interrupt. When the erase cycle terminates, the ER bit will be automatically cleared to zero by the microcontroller, informing the user that the page data has been erased. The application program can therefore poll the ER bit to determine when the erase cycle has ended. After the erase operation is finished, the EREN bit will be set low by hardware. The Data EEPROM erased page content will be zero after a page erase operation.

Writing Operation to the EEPROM

Writing data to the EEPROM can be implemented by two modes for this device, byte write mode or page write mode, which is controlled by the EEPROM operation mode selection bit, MODE, in the EEC register.

Byte Write Mode

The EEPROM byte write operation can be executed when the mode selection bit, MODE, is cleared to zero. For byte write operations the desired EEPROM address should first be placed in the EEAH and EEAL registers and the data to be written should be 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 operation. After this, the WR bit in the EEC register must be immediately set high to initiate the EEPROM write cycle. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing any write operation and then set again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not 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 end. After the write operation is finished, the WREN bit will be set low by hardware. Note that a byte erase operation will automatically be executed before a byte write operation is successfully activated.

Page Write Mode

Before a page write operation is executed, it is important to ensure that a relevant page erase operation has been successfully executed. The EEPROM page write operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page write. The internal page buffer will be cleared by hardware after power-on reset. When the EEPROM write enable control bit, namely WREN, is changed from "1" to "0", the internal page buffer will also be cleared. Note that when the WREN bit is changed from "0" to "1", the internal page buffer will not be cleared. A page write is initiated in the same way as a byte write initiation except that the EEPROM data can be written up to 16 bytes. The EEPROM address higher 5 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page write operation mode the lower 4-bit address value will automatically be incremented by one after each data byte is written into the EED register. However, the higher 5-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, namely 0FH, the EEPROM address low 4-bit value will stop at 0FH. The EEPROM address will not "roll over". At this point any data write operations to the EED register will be invalid.

For page write operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers and the data to be written is placed in the EED registers. The maximum data length is 16-byte. Note that when the data byte is written into the EED register, then the data in the EED register will be loaded into the internal page buffer and the current address value will automatically be incremented by one. When the page data is completely written into the page buffer, then the WREN bit in the EEC register should be set high to enable write operations and the WR bit must be immediately set high to initiate the EEPROM erase process. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing any write operation and then set again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not 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 end. After the write operation is finished, the WREN bit will be set low by hardware.

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 interrupt is generated when an EEPROM erase or write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. When an EEPROM erase or write cycle ends, the DEF request flag will be set high. If the global and EEPROM interrupts are enabled and the stack is not full, a jump to the EEPROM Interrupt vector will take place. When the interrupt is serviced, the EEPROM interrupt flag will be automatically reset. More details can be obtained in the Interrupt section.

Programming Considerations

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

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

Programming Examples

Reading a Data Byte from the EEPROM – Polling Method

```
MOV  A, 40H                ; set memory pointer low byte MP1L
MOV  MP1L, A               ; MP1 points to EEC register
MOV  A, 01H                ; set memory pointer high byte MP1H
MOV  MP1H, A
CLR  IAR1.4                ; clear MODE bit, select byte operation mode
MOV  A, EEPROM_ADRES_H     ; user defined high byte address
MOV  EEAH, A
MOV  A, EEPROM_ADRES_L     ; user defined low byte address
MOV  EEAL, 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 function
CLR  MP1H
MOV  A, EED                ; move read data to register
MOV  READ_DATA, A
```

Reading a Data Page from the EEPROM – Polling Method

```
MOV  A, 40H                 ; set memory pointer low byte MP1L
MOV  MP1L, A                ; MP1 points to EEC register
MOV  A, 01H                 ; set memory pointer high byte MP1H
MOV  MP1H, A
SET  IAR1.4                 ; set MODE bit, select page operation mode
MOV  A, EEPROM_ADRES_H      ; user defined high byte address
MOV  EEAH, A
MOV  A, EEPROM_ADRES_L      ; user defined low byte address
MOV  EEAL, A
SET  IAR1.1                 ; set RDEN bit, enable read operations
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL READ
CALL READ
:
:
JMP  PAGE_READ_FINISH
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
READ:
SET  IAR1.0                 ; start Read Cycle - set RD bit
BACK:
SZ   IAR1.0                 ; check for read cycle end
JMP  BACK
MOV  A, EED                 ; move read data to register
MOV  READ_DATA, A
RET
:
PAGE_READ_FINISH
CLR  IAR1                   ; disable EEPROM read function
CLR  MP1H
```

Erasing a Data Page to the EEPROM – Polling Method

```
MOV  A, 40H                 ; set memory pointer low byte MP1L
MOV  MP1L, A                ; MP1 points to EEC register
MOV  A, 01H                 ; set memory pointer high byte MP1H
MOV  MP1H, A
SET  IAR1.4                 ; set MODE bit, select page operation mode
MOV  A, EEPROM_ADRES_H      ; user defined high byte address
MOV  EEAH, A
MOV  A, EEPROM_ADRES_L      ; user defined low byte address
MOV  EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP  Erase_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV  A, EEPROM_DATA         ; user define data, erase mode don’t care data value
MOV  EED, A
RET
:
Erase_START:
CLR  EMI
SET  IAR1.6                 ; set EREN bit, select erase operations
SET  IAR1.5                 ; start Erase Cycle - set ER bit - executed immediately
                            ; after setting EREN bit
SET  EMI
```

```
BACK:
SZ   IAR1.5                  ; check for erase cycle end
JMP  BACK
CLR  MP1H
```

Writing a Data Byte to the EEPROM – polling method

```
MOV  A, 40H                  ; set memory pointer low byte MP1L
MOV  MP1L, A                 ; MP1 points to EEC register
MOV  A, 01H                  ; set memory pointer high byte MP1H
MOV  MP1H, A
CLR  IAR1.4                  ; clear MODE bit, select byte write mode
MOV  A, EEPROM_ADRES_H       ; user defined high byte address
MOV  EEAH, A
MOV  A, EEPROM_ADRES_L       ; user defined low byte address
MOV  EEAL, A
MOV  A, EEPROM_DATA          ; user define data
MOV  EED, A
CLR  EMI
SET  IAR1.3                  ; set WREN bit, enable write operations
SET  IAR1.2                  ; start Write Cycle - set WR bit - executed immediately
                             ; after setting WREN bit
SET  EMI
BACK:
SZ   IAR1.2                  ; check for write cycle end
JMP  BACK
CLR  MP1H
```

Writing a Data Page to the EEPROM – Polling Method

```
MOV  A, 40H                  ; set memory pointer low byte MP1L
MOV  MP1L, A                 ; MP1 points to EEC register
MOV  A, 01H                  ; set memory pointer high byte MP1H
MOV  MP1H, A
SET  IAR1.4                  ; set MODE bit, select page operation mode
MOV  A, EEPROM_ADRES_H       ; user defined high byte address
MOV  EEAH, A
MOV  A, EEPROM_ADRES_L       ; user defined low byte address
MOV  EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP  WRITE_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV  A, EEPROM_DATA          ; user define data
MOV  EED, A
RET
:
WRITE_START:
CLR  EMI
SET  IAR1.3                  ; set WREN bit, enable write operations
SET  IAR1.2                  ; start Write Cycle - set WR bit - executed immediately
                             ; after setting WREN bit
SET  EMI
BACK:
SZ   IAR1.2                  ; check for write cycle end
JMP  BACK
CLR  MP1H
```

Oscillators

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

Oscillator Overview

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

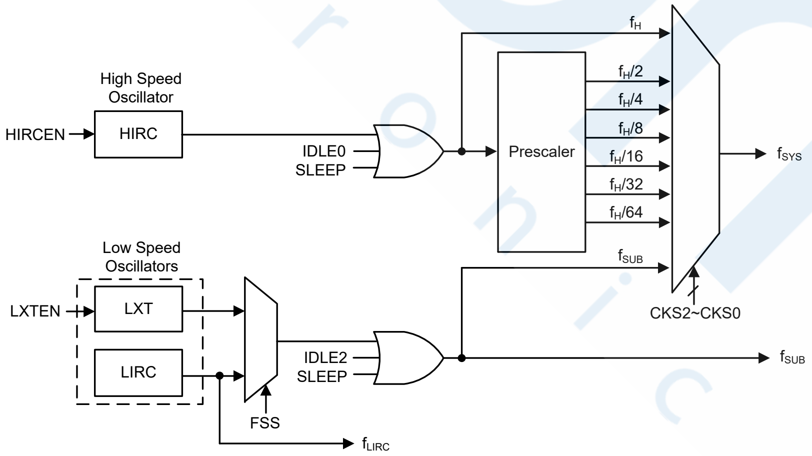
Type	Name	Frequency	Pins
Internal High Speed RC	HIRC	8/12/16MHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

System Clock Configurations

There are several oscillator sources, one high speed oscillator and two low speed oscillators. The high speed system clock is sourced from the internal 8/12/16MHz RC oscillator, HIRC. The low speed oscillators are the external 32.768kHz crystal oscillator, LXT, and the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

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



System Clock Configurations

Internal High Speed RC Oscillator – HIRC

The internal high speed RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 8MHz, 12MHz and 16MHz, which are selected by HIRC1~HIRC0 bits in the HIRCC register. These bits must be setup to match the selected configuration option frequency to ensure that the HIRC frequency accuracy specificied in the A.C. Characterisitics is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that if this internal system clock option is selected, it requires no external pins for its operation.

External 32.768kHz Crystal Oscillator – LXT

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

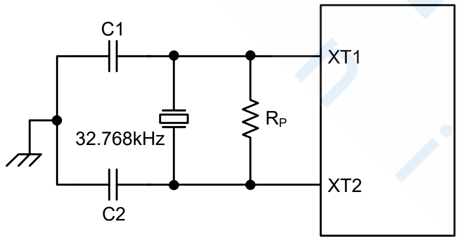
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE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

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

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

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

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



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

External LXT Oscillator

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

32.768kHz Crystal Recommended Capacitor Values

LXT Oscillator Low Power Function

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

LXTSP	LXT Mode
0	Low Power
1	Speed Up

When the LXTSP bit is set to high, the LXT Speed Up Mode will be enabled. In the Speed Up Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up, it can be placed into the Low Power Mode by clearing the LXTSP bit to zero and the oscillator will continue to run but with reduced current consumption. It is important to note that the LXT operating mode switching must be properly controlled before the LXT oscillator clock is selected as the system clock source. Once the LXT oscillator clock is selected as the system clock source using the CKS2~CKS0 bits and FSS bit in the SCC register, the LXT oscillator operating mode cannot be changed.

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

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is one of the low frequency oscillator choices, which is selected via a software control bit, FSS. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

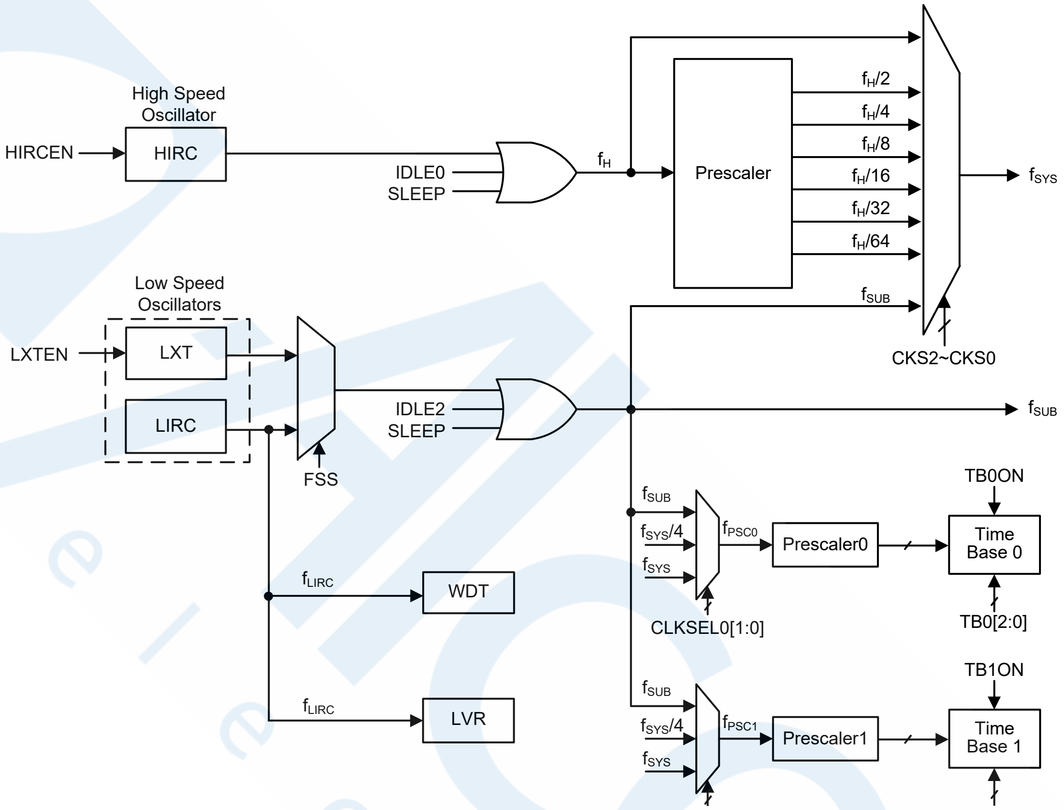
Operating Modes and System Clocks

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

System Clocks

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

The main system clock can come from a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock 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 by either the LXT or LIRC oscillator, selected via configuring the FSS bit in the SCC register. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.



Device Clock Configurations

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

System Operation Modes

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

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

Operation Mode	CPU	Register Setting			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
IDLE2	Off	1	0	000~110	On	On	Off	On
				111	Off			
SLEEP	Off	0	0	xxx	Off	Off	Off	On $^{(2)}$

"x": Don't care

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

FAST Mode

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

SLOW Mode

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

SLEEP Mode

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

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

The registers, SCC, HIRCC 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	—	—	FSS	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
LXTC	—	—	—	—	—	LXTSP	LXTF	LXTEN

System Operating Mode Control Register List

• SCC Register

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

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

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

Bit 4~3 Unimplemented, read as “0”.

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

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

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

Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits or FSS bit. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.

Clock switching delay time = $4\times t_{SYS}+[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$, where $t_{Curr.}$ indicates the current clock period, $t_{Tar.}$ indicates the target clock period and t_{SYS} indicates the current system clock period.

• 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: 8MHz
- 01: 12MHz
- 10: 16MHz
- 11: 8MHz

When the HIRC oscillator is enabled or the HIRC frequency selection is changed by application program, the clock frequency will automatically be changed after the HIRCF flag is set high.

It is recommended that the HIRC frequency selected by these two bits should be same with the frequency determined by the configuration option 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, 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	—	—	—	—	—	LXTSP	LXTF	LXTEN
R/W	—	—	—	—	—	R/W	R	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LXTSP**: LXT speed up control
- 0: Disable – Low power
- 1: Enable – Speed up

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

Bit 1 **LXTF**: LXT oscillator stable flag
- 0: LXT unstable
- 1: LXT stable

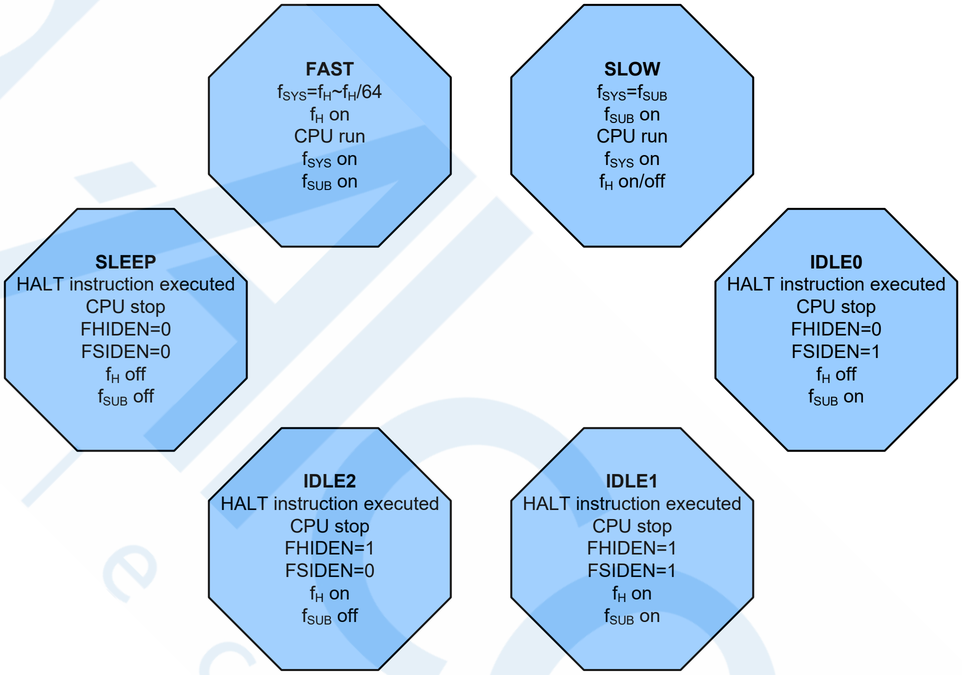
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set to 1 to enable the LXT oscillator, the LXTF bit will first be cleared to 0 and then set to 1 after the LXT oscillator is stable.

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

Operating Mode Switching

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

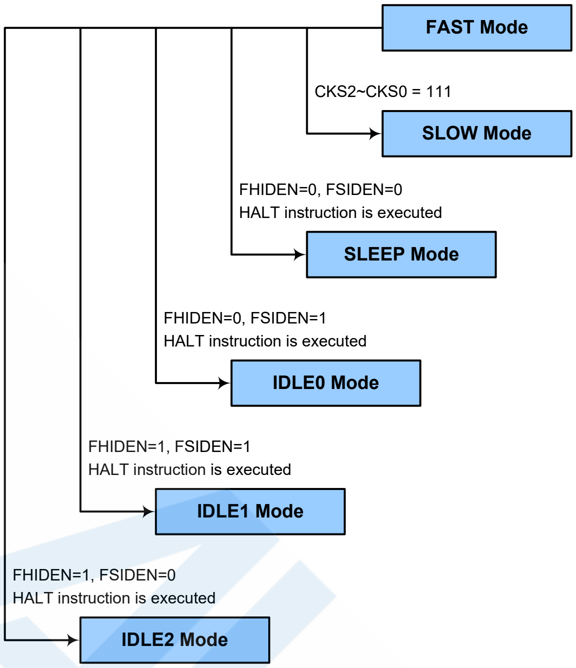
In simple terms, mode switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



FAST Mode to SLOW Mode Switching

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

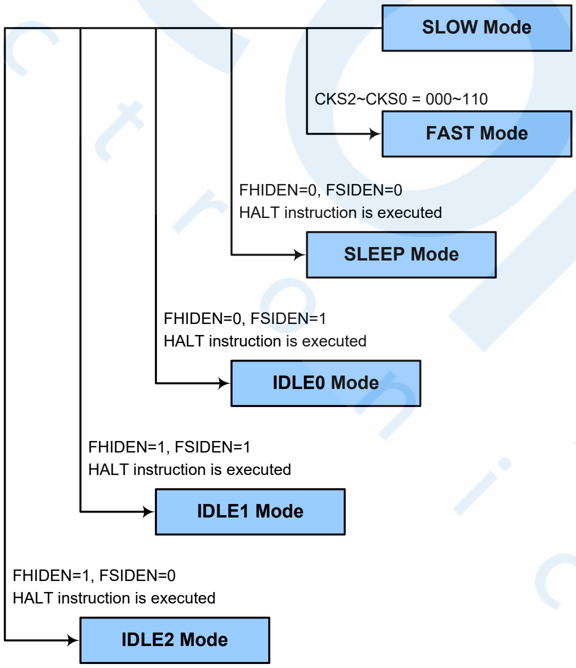
The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires the selected oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

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

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



Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

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

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

Entering the IDLE2 Mode

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

- The f_H clock will be on but the f_{SUB} clock will be off and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.

- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting as the WDT function is always enabled.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These must either be set as outputs or if set 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 set as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the LXT or LIRC oscillator has enabled.

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

Wake-up

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

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

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

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

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

Watchdog Timer

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

Watchdog Timer Clock Source

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

Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period as well as the Watchdog Timer the enable and the MCU reset operation.

- **WDTC Register**

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

Bit 7~3 **WE4~WE0**: WDT function software control
01010 or 10101: Enable
Other values: Reset MCU
When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set high.

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

- **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

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

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

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

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

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to offer the enable control of the Watchdog Timer and the MCU reset. The WDT function will be enabled when the WE4~WE0 bits are equal to 10101B or 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.

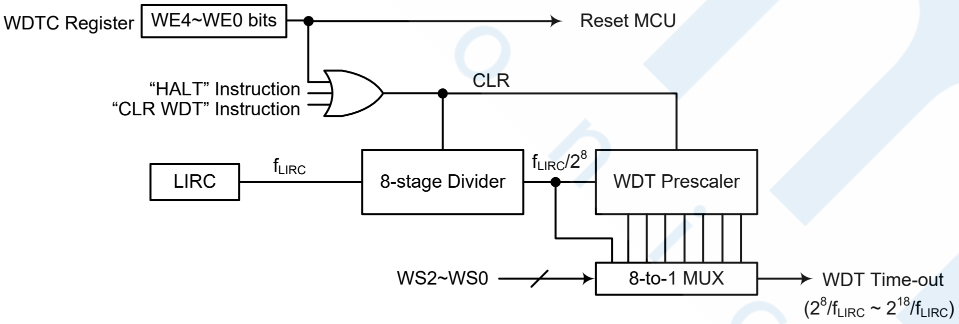
WE4~WE0 Bits	WDT Function
10101B or 01010B	Enable
Any other value	Reset MCU

Watchdog Timer Enable/Reset Control

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

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

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

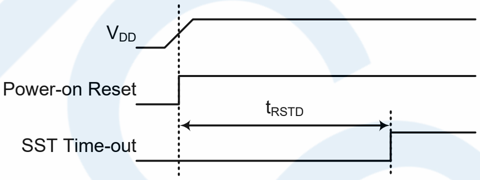
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

Reset Functions

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

Power-on Reset

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



Power-on Reset Timing Chart

Internal Reset Control

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

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

Internal Reset Function Control

- **RSTC Register**

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

Bit 7~0 **RSTC7~RSTC0**: Reset function control
01010101: No operation
10101010: No operation
Other values: Reset MCU

If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET}, and the RSTF bit in the RSTFC register will be set to 1. All resets will reset this register to POR value except the WDT time out hardware warm reset.

- **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
0: Not occurred
1: Occurred

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

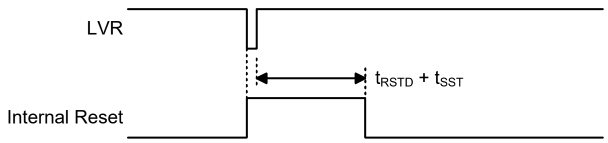
Bit 2 **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.

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

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

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provide an MCU reset should the value fall below a certain predefined level. The LVR function can be enabled or disabled by the LVRC control register. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set 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 LVR/LVD Electrical characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set 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 or 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 selection

01100110: 1.7V
01010101: 1.9V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
11110000: LVR disable
Any other value: Generates MCU reset – register is reset to POR value

When an actual low voltage condition as specified above occurs, 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 six defined register values above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time, t_{SRESET}. However in this situation the register contents will be reset to the POR value.

• RSTFC Register

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

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

Refer to the Internal Reset Control section.

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

0: Not occurred
1: Occurred

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

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

0: Not occurred
1: Occurred

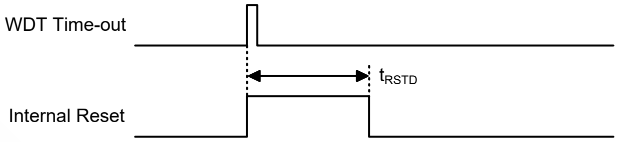
This bit is set high if the LVRC register contains any non-defined register values. This in effect acts like a software reset function. This bit can only be cleared to zero by the application program.

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

Refer to the Watchdog Timer Control Register section.

Watchdog Time-out Reset during Normal Operation

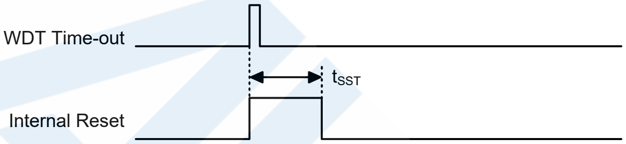
When the Watchdog time-out Reset during normal operations in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to “1”.



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



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

Reset Initial Conditions

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

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

“u”: 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	Cleared 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	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	---- xxxx	---- uuuu	---- uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- 0x00	---- uuuu	---- uuuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	--00 --00	--00 --00	--uu --uu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
SLEDC0	0000 0000	0000 0000	uuuu uuuu
SLEDC1	--00 0000	--00 0000	--uu uuuu
TB0C	0--- -000	0--- -000	u--- -uuu
PSC0R	---- --00	---- --00	---- --uu
LVRC	0110 0110	0110 0110	uuuu uuuu
LVPUC	---- ---0	---- ---0	---- ---u
LXTC	---- -000	---- -000	---- -uuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
WDTC	0101 0011	0101 0011	uuuu uuuu
IICC0	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	0000 000-	0000 000-	uuuu uuu-
IICTOC	0000 0000	0000 0000	uuuu uuuu
RSTC	0101 0101	0101 0101	uuuu uuuu
USR	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	uuuu uuuu

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
UCR3	---- ---0	---- ---0	---- ---u
BRDH	0000 0000	0000 0000	uuuu uuuu
BRDL	0000 0000	0000 0000	uuuu uuuu
UFCR	--00 0000	--00 0000	--uu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
RxCNT	---- -000	---- -000	---- -uuu
IFS0	-000 0000	-000 0000	-uuu uuuu
SCC	000- -000	000- -000	uuu- -uuu
LVDC	--00 -000	--00 -000	--uu -uuu
HIRCC	---- 0001	---- 0001	---- uuuu
PC	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	uuuu uuuu
MFI0	--00 --00	--00 --00	--uu --uu
PD	---- --11	---- --11	---- --uu
PDC	---- --11	---- --11	---- --uu
PDPU	---- --00	---- --00	---- --uu
EEAL	0000 0000	0000 0000	uuuu uuuu
EEAH	---- ---0	---- ---0	---- ---u
EED	0000 0000	0000 0000	uuuu uuuu
TKTMR	0000 0000	0000 0000	uuuu uuuu
TKC0	-000 0000	-000 0000	-uuu uuuu
TL16DL	0000 0000	0000 0000	uuuu uuuu
TK16DH	0000 0000	0000 0000	uuuu uuuu
TKC1	---- --11	---- --11	---- --uu
TKM016DL	0000 0000	0000 0000	uuuu uuuu
TKM016DH	0000 0000	0000 0000	uuuu uuuu
TKM0ROL	0000 0000	0000 0000	uuuu uuuu
TKM0ROH	---- --00	---- --00	---- --uu
TKM0C0	0000 0000	0000 0000	uuuu uuuu
TKM0C1	0-00 0000	0-00 0000	u-uu uuuu
TKM116DL	0000 0000	0000 0000	uuuu uuuu
TKM116DH	0000 0000	0000 0000	uuuu uuuu
TKM1ROL	0000 0000	0000 0000	uuuu uuuu
TKM1ROH	---- --00	---- --00	---- --uu
TKM1C0	0000 0000	0000 0000	uuuu uuuu
TKM1C1	0-00 0000	0-00 0000	u-uu uuuu
TKM216DL	0000 0000	0000 0000	uuuu uuuu
TKM216DH	0000 0000	0000 0000	uuuu uuuu
TKM2ROL	0000 0000	0000 0000	uuuu uuuu
TKM2ROH	---- --00	---- --00	---- --uu
TKM2C0	0000 0000	0000 0000	uuuu uuuu
TKM2C1	0-00 0000	0-00 0000	u-uu uuuu
TKM316DL	0000 0000	0000 0000	uuuu uuuu
TKM316DH	0000 0000	0000 0000	uuuu uuuu
TKM3ROL	0000 0000	0000 0000	uuuu uuuu
TKM3ROH	---- --00	---- --00	---- --uu

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TKM3C0	0000 0000	0000 0000	uuuu uuuu
TKM3C1	0-00 0000	0-00 0000	u-uu uuuu
ORMC	0000 0000	0000 0000	0000 0000
CTM0C0	0000 0000	0000 0000	uuuu uuuu
CTM0C1	0000 0000	0000 0000	uuuu uuuu
CTM0DL	0000 0000	0000 0000	uuuu uuuu
CTM0DH	---- --00	---- --00	---- --uu
CTM0AL	0000 0000	0000 0000	uuuu uuuu
CTM0AH	---- --00	---- --00	---- --uu
CTM1C0	0000 0000	0000 0000	uuuu uuuu
CTM1C1	0000 0000	0000 0000	uuuu uuuu
CTM1DL	0000 0000	0000 0000	uuuu uuuu
CTM1DH	---- --00	---- --00	---- --uu
CTM1AL	0000 0000	0000 0000	uuuu uuuu
CTM1AH	---- --00	---- --00	---- --uu
INTEG	---- --00	---- --00	---- --uu
PSC1R	---- --00	---- --00	---- --uu
TB1C	0--- -000	0--- -000	u--- -uuu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 0000	0000 0000	uuuu uuuu
PBS1	0000 0000	0000 0000	uuuu uuuu
PCS0	0000 0000	0000 0000	uuuu uuuu
PCS1	0000 0000	0000 0000	uuuu uuuu
PDS0	---- 0000	---- 0000	---- uuuu
PANS	0000 0000	0000 0000	uuuu uuuu
PCNS	0000 0000	0000 0000	uuuu uuuu
PDNS	---- --00	---- --00	---- --uu
PTM0C0	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	uuuu uuuu
PTM0DL	0000 0000	0000 0000	uuuu uuuu
PTM0DH	---- --00	---- --00	---- --uu
PTM0AL	0000 0000	0000 0000	uuuu uuuu
PTM0AH	---- --00	---- --00	---- --uu
PTM0RPL	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	---- --00	---- --00	---- --uu
SLCDC0	0000 ----	0000 ----	uuuu ----
SLCDS0	0000 0000	0000 0000	uuuu uuuu
SLCDS1	0000 0000	0000 0000	uuuu uuuu
SLCDS2	0000 0000	0000 0000	uuuu uuuu
SLCDS3	---- --00	---- --00	---- --uu
EEC	0000 0000	0000 0000	uuuu uuuu

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

Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA~PD. These I/O ports are mapped to the RAM Data Memory w108ith specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction “MOV A, [m]”, where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	—	—	—	—	—	—	PD1	PD0
PDC	—	—	—	—	—	—	PDC1	PDC0
PDPU	—	—	—	—	—	—	PDPU1	PDPU0
LVPUC	—	—	—	—	—	—	—	LVPU

“—”: Unimplemented, read as “0”

I/O Logic Function Register List

Pull-high Resistors

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

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

• PxPU Register

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

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

The PxPUn bit is used to control the pin pull-high function. Here the "x" can be A, B, C or D. 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 selection for low voltage power supply
0: All pin pull-high resistors are 60kΩ @ 3V
1: All pin pull-high resistors are 15kΩ @ 3V

The LVPUC register is used to select the pull-high resistor value for low voltage power supply applications. Note that the LVPU bit is only available when the corresponding pin pull-high function is enabled. This bit will have no effect on selecting the pull-high resistor value 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 register only when the pin is selected as a general purpose input and the MCU enters the IDLE or SLEEP mode.

- **PAWU Register**

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

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

I/O Port Control Registers

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

- **PxC Register**

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

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

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

I/O Port Source Current Selection

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLEDC0	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
SLEDC1	—	—	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10

I/O Port Source Current Selection Register List

- **SLEDC0 Register**

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

Bit 7~6 **SLEDC07~SLEDC06**: PB7~PB4 source current selection
00: Source current = Level 0 (Min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (Max.)

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

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

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

- **SLEDC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **SLEDC15~SLEDC14**: PD1~PD0 source current selection
00: Source current = Level 0 (Min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (Max.)

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

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

I/O Port Sink Current Selection

The device supports different output sink current driving capability for PA, PC and PD ports. With the selection registers, PxNS, specific I/O port can support two levels of the sink current driving capability. These sink current selection bits are available when the corresponding pin is configured as a CMOS output. Otherwise, these select bits have no effect. Users should refer to the Input/Output Characteristics section to select the desired output sink current for different applications.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PANS	PANS7	PANS6	PANS5	PANS4	PANS3	PANS2	PANS1	PANS0
PCNS	PCNS7	PCNS6	PCNS5	PCNS4	PCNS3	PCNS2	PCNS1	PCNS0
PDNS	—	—	—	—	—	—	PDNS1	PDNS0

I/O Port Source Current Selection Register List

- **PANS Register**

Bit	7	6	5	4	3	2	1	0
Name	PANS7	PANS6	PANS5	PANS4	PANS3	PANS2	PANS1	PANS0
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 **PANS7**: PA7 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 6 **PANS6**: PA6 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 5 **PANS5**: PA5 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 4 **PANS4**: PA4 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 3 **PANS3**: PA3 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 2 **PANS2**: PA2 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 1 **PANS1**: PA1 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 0 **PANS0**: PA0 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

- **PCNS Register**

Bit	7	6	5	4	3	2	1	0
Name	PCNS7	PCNS6	PCNS5	PCNS4	PCNS3	PCNS2	PCNS1	PCNS0
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 **PCNS7**: PC7 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 6 **PCNS6**: PC6 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 5 **PCNS5**: PC5 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 4 **PCNS4**: PC4 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 3 **PCNS3**: PC3 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 2 **PCNS2**: PC2 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 1 **PCNS1**: PC1 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 0 **PCNS0**: PC0 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

- **PDNS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PDNS1	PANS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1 **PDNS1**: PD1 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Bit 0 **PDNS0**: PD0 sink current selection (NMOS adjust)
0: Sink current = Level 0 (Min.)
1: Sink current = Level 1 (Max.)

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	—	—	—	—	PDS03	PDS02	PDS01	PDS00
IFS0	—	SDAPS	SCLPS	RXPS	PTCK0PS1	PTCK0PS0	CTCK0PS1	CTCK0PS0

Pin-shared Function Selection Register List

• PAS0 Register

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

Bit 7~6 **PAS07~PAS06**: PA3 Pin-Shared function selection
00: PA3
01: SSEG3/SCOM3
10: SDA
11: RX/TX

Bit 5~4 **PAS05~PAS04**: PA2 Pin-Shared function selection
00: PA2
01: SSEG2/SCOM2
10: PA2
11: PA2

Bit 3~2 **PAS03~PAS02**: PA1 Pin-Shared function selection
00: PA1/CTCK0
01: SSEG1/SCOM1
10: CTP0B
11: PA1/CTCK0

Bit 1~0 **PAS01~PAS00**: PA0 Pin-Shared function selection
00: PA0/PTCK0
01: SSEG0/SCOM0
10: PA0/PTCK0
11: PA0/PTCK0

• PAS1 Register

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

Bit 7~6 **PAS17~PAS16**: PA7 Pin-Shared function selection
00: PA7
01: SSEG7/SCOM7
10: SCL
11: TX

Bit 5~4 **PAS15~PAS14**: PA6 Pin-Shared function selection
00: PA6
01: SSEG6/SCOM6
10: PTP0
11: XT2

Bit 3~2 **PAS13~PAS12**: PA5 Pin-Shared function selection
00: PA5
01: SSEG5/SCOM5
10: CTP0
11: XT1

Bit 1~0 **PAS11~PAS10**: PA4 Pin-Shared function selection
00: PA4/INT0/CTCK0
01: SSEG4/SCOM4
10: PTP0
11: PA4/INT0/CTCK0

- **PBS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
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 **PBS07~PBS06**: PB3 Pin-Shared function selection
00: PB3
01: SSEG11/SCOM11
10: KEY4
11: PB3

Bit 5~4 **PBS05~PBS04**: PB2 Pin-Shared function selection
00: PB2
01: SSEG10/SCOM10
10: KEY3
11: PB2

Bit 3~2 **PBS03~PBS02**: PB1 Pin-Shared function selection
00: PB1
01: SSEG9/SCOM9
10: KEY2
11: PB1

Bit 1~0 **PBS01~PBS00**: PB0 Pin-Shared function selection
00: PB0
01: SSEG8/SCOM8
10: KEY1
11: PB0

- **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PBS17~PBS16**: PB7 Pin-Shared function selection
00: PB7/CTCK0
01: SSEG15/SCOM15
10: KEY8
11: CTP0

Bit 5~4 **PBS15~PBS14**: PB6 Pin-Shared function selection
00: PB6
01: SSEG14/SCOM14
10: KEY7
11: CTP0B

Bit 3~2 **PBS13~PBS12**: PB5 Pin-Shared function selection
00: PB5
01: SSEG13/SCOM13
10: KEY6
11: PB5

Bit 1~0 **PBS11~PBS10**: PB4 Pin-Shared function selection
00: PB4
01: SSEG12/SCOM12
10: KEY5
11: PB4

- **PCS0 Register**

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

Bit 7~6 **PCS07~PCS06**: PC3 Pin-Shared function selection
00: PC3
01: SSEG19/SCOM19
10: KEY12
11: SDA

Bit 5~4 **PCS05~PCS04**: PC2 Pin-Shared function selection
00: PC2
01: SSEG18/SCOM18
10: KEY11
11: SCL

Bit 3~2 **PCS03~PCS02**: PC1 Pin-Shared function selection
00: PC1
01: SSEG17/SCOM17
10: KEY10
11: TX

Bit 1~0 **PCS01~PCS00**: PC0 Pin-Shared function selection
00: PC0
01: SSEG16/SCOM16
10: KEY9
11: RX/TX

- **PCS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PCS17~PCS16**: PC7 Pin-Shared function selection
00: PC7/PTCK0
01: SSEG23/SCOM23
10: KEY16
11: CTP0

Bit 5~4 **PCS15~PCS14**: PC6 Pin-Shared function selection
00: PC6
01: SSEG22/SCOM22
10: KEY15
11: PTP0

Bit 3~2 **PCS13~PCS12**: PC5 Pin-Shared function selection
00: PC5/PTCK0
01: SSEG21/SCOM21
10: KEY14
11: CTP0B

Bit 1~0 **PCS11~PCS10**: PC4 Pin-Shared function selection
00: PC4
01: SSEG20/SCOM20
10: KEY13
11: PTP0B

- **PDS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PDS03	PDS02	PDS01	PDS00
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 **PDS03~PDS02**: PD1 Pin-Shared function selection
00: PD1
01: SSEG25/SCOM25
10: CTP1B
11: PD1

Bit 1~0 **PDS01~PDS00**: PD0 Pin-Shared function selection
00: PD0/CTCK1
01: SSEG24/SCOM24
10: CTP1
11: PD0/CTCK1

- **IFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	SDAPS	SCLPS	RXPS	PTCK0PS1	PTCK0PS0	CTCK0PS1	CTCK0PS0
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 **SDAPS**: SDA input source pin selection
0: PA3
1: PC3

Bit 5 **SCLPS**: SCL input source pin selection
0: PA7
1: PC2

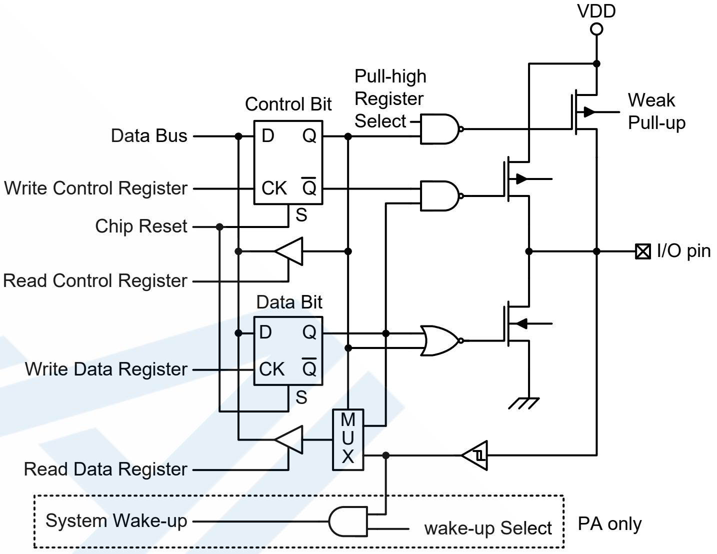
Bit 4 **RXPS**: RX/TX input source pin selection
0: PA3
1: PC0

Bit 3~2 **PTCK0PS1~PTCK0PS0**: PTCK0 input source pin selection
00: PA0
01: PC5
10: PC7
11: PA0

Bit 1~0 **CTCK0PS1~CTCK0PS0**: CTCK0 input source pin selection
00: PA4
01: PA1
10: PB7
11: PA4

I/O Pin Structures

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



Logic Function Input/Output Structure

Programming Considerations

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

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

Timer Modules – TM

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

The common features of the different TM types are described here with more detailed information provided in the individual Compact and Periodic 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 Periodic Type TM. Although similar in nature, the different TM types vary in their feature complexity. The common features to all of the Compact and Periodic TMs will be described in this section. The detailed operation regarding each of the TM types will be described in separate sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

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

TM Function Summary

TM Operation

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

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTnCK2~xTnCK0 bits in the xTMn control registers, where “x” stands for C or P type TM and “n” stands for the specific TM serial number. The clock source can be a ratio of the system clock f_{SYS} or the internal high clock f_H, the f_{SUB} clock source or the external xTCKn pin. The xTCKn pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

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

TM External Pins

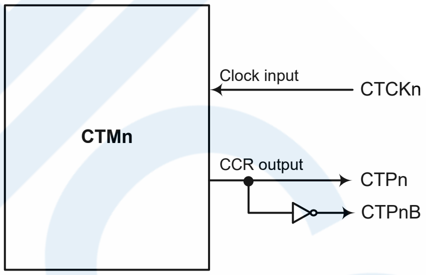
Each of the TMs, irrespective of what type, has one input pin with the label xTCKn. The xTMn input pin, xTCKn, is essentially a clock source for the xTMn and is selected using the xTnCK2~xTnCK0 bits in the xTMnC0 register. This external TM input pin allows an external clock source to drive the internal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

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

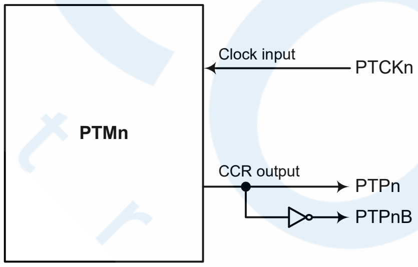
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be selected using the relevant pin-shared function selection bits described in the Pin-shared Function section.

CTM		PTM	
Input	Output	Input	Output
CTCK0; CTCK1	CTP0, CTP0B; CTP1, CTP1B	PTCK0	PTP0, PTP0B

TM External Pins



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

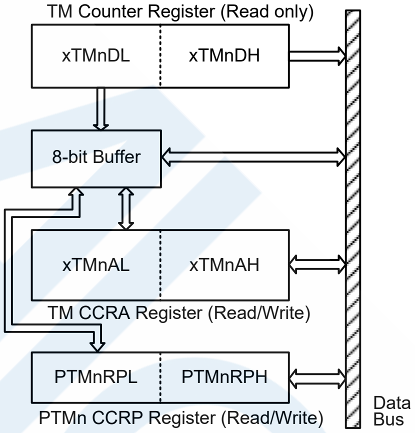


PTM Function Pin Block Diagram (n=0)

Programming Considerations

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

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

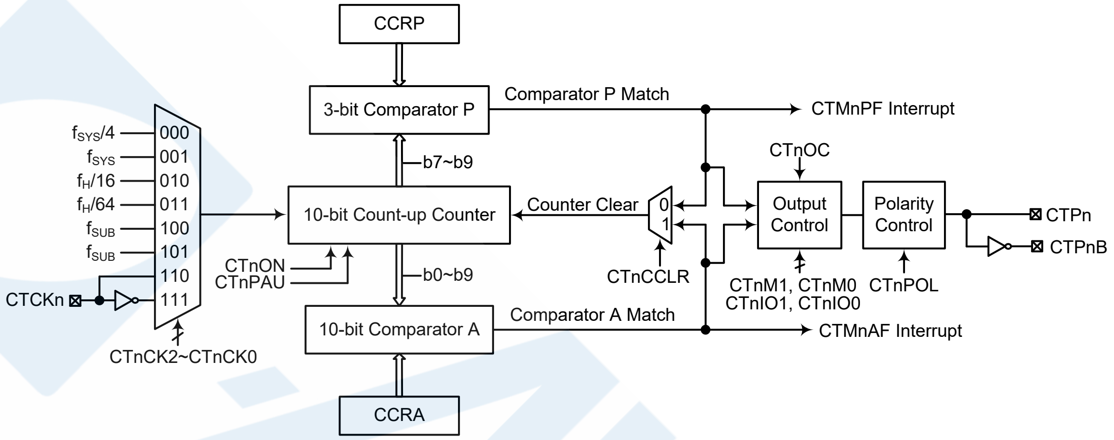


The following steps show the read and write procedures:

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

Compact Type TM – CTM

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



Note: 1. As the CTMn external pins are pin-shared with other functions, so before using the CTMn function the relevant pin-shared function registers must be set properly to enable the CTMn pin function. The CTCKn pins, if used, must also be set as an imput by setting the corresponding bits in the port control register.

2. The CTPnB is the inverted signal of the CTPn.

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

Compact Type TM Operation

The size of Compact TM is 10-bit wide ande 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 3 bits in the counter while the CCRA is the 10 bits and therefore compares with all counter bits.

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

Compact Type TM Register Description

Overall operation of the Compact 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 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 CTMn external pin 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 CTnMC1 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 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 CTMn 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 **D7~D0**: 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 **D9~D8**: 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 **D7~D0**: 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 **D9~D8**: CTMn CCRA High Byte Register bit 1 ~ bit 0
CTMn 10-bit CCRA bit 9 ~ bit 8

Compact Type TM Operating Modes

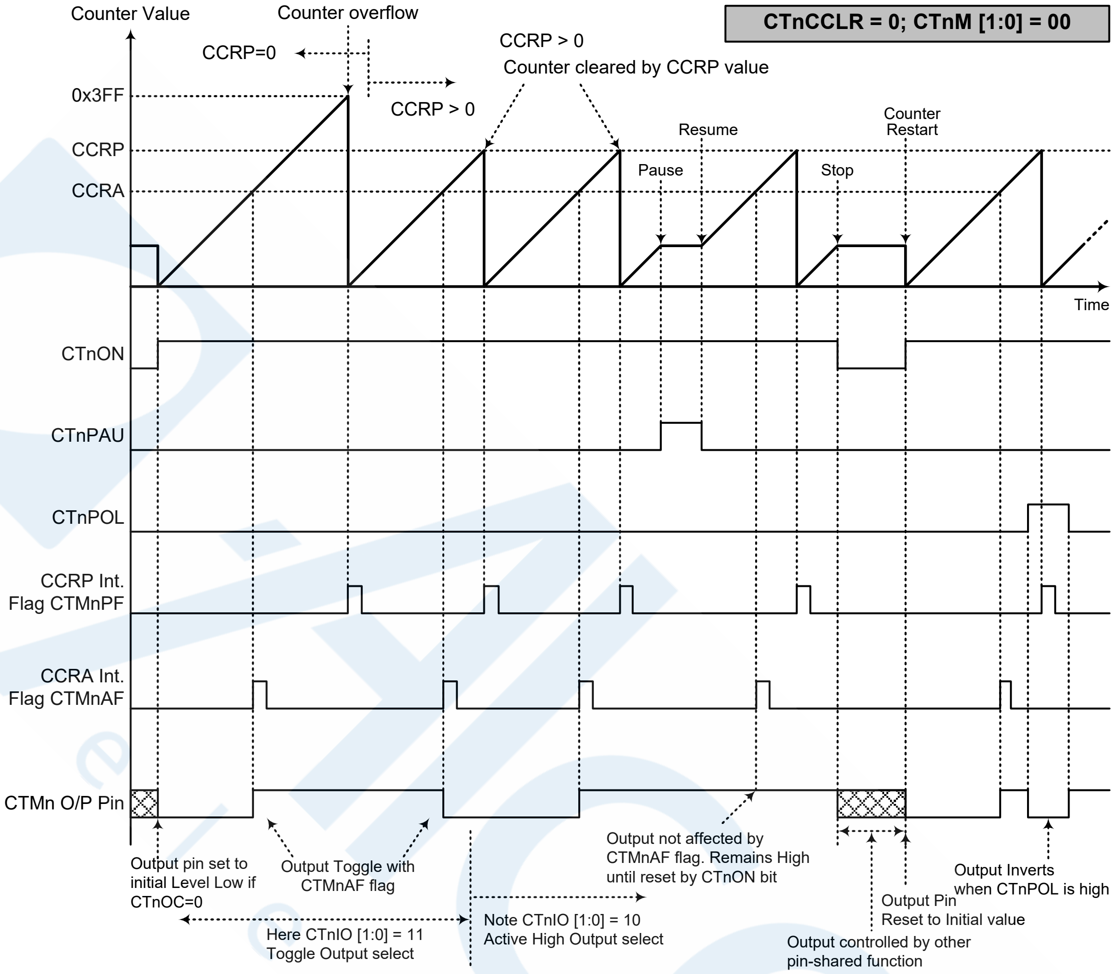
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

Compare Match Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTnMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMnAF and CTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

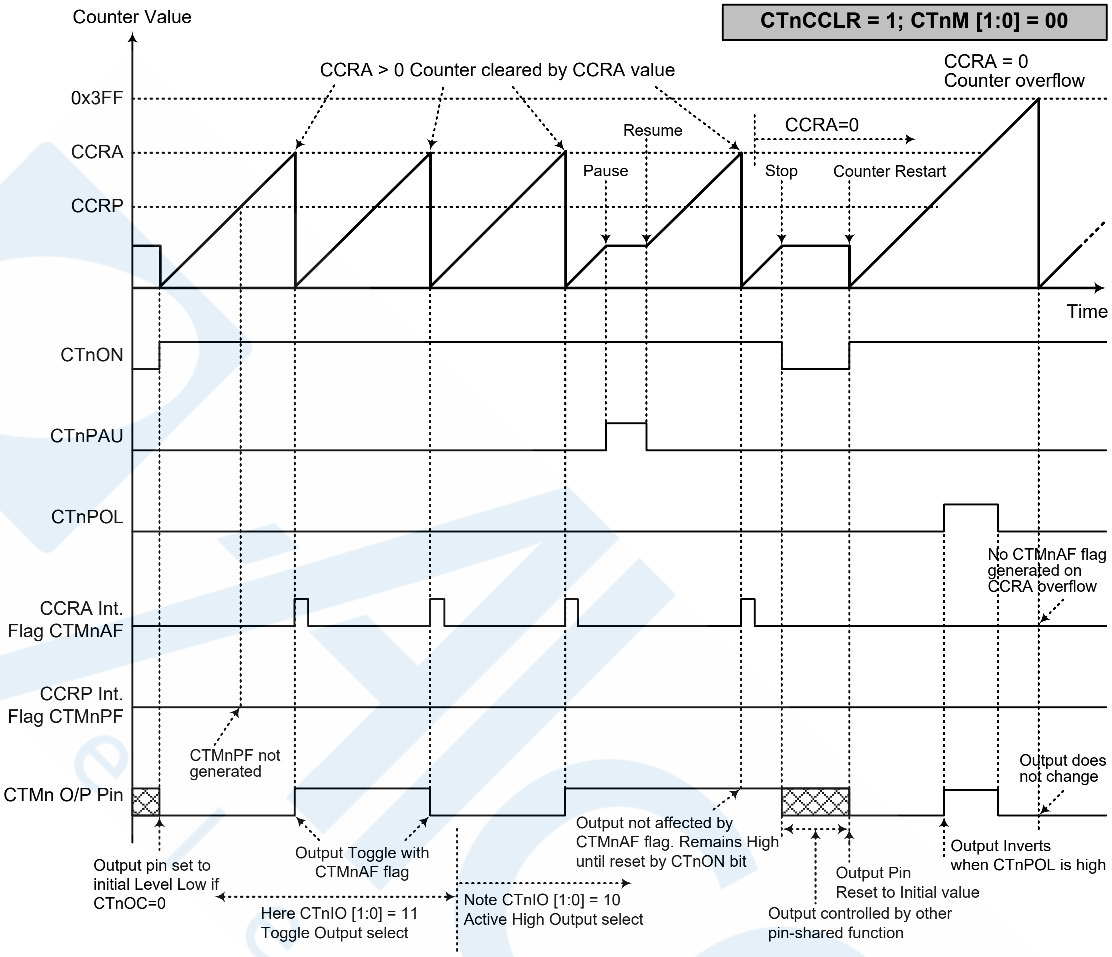
If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is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

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



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

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



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

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

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.

- **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=100b, CCRA=128,

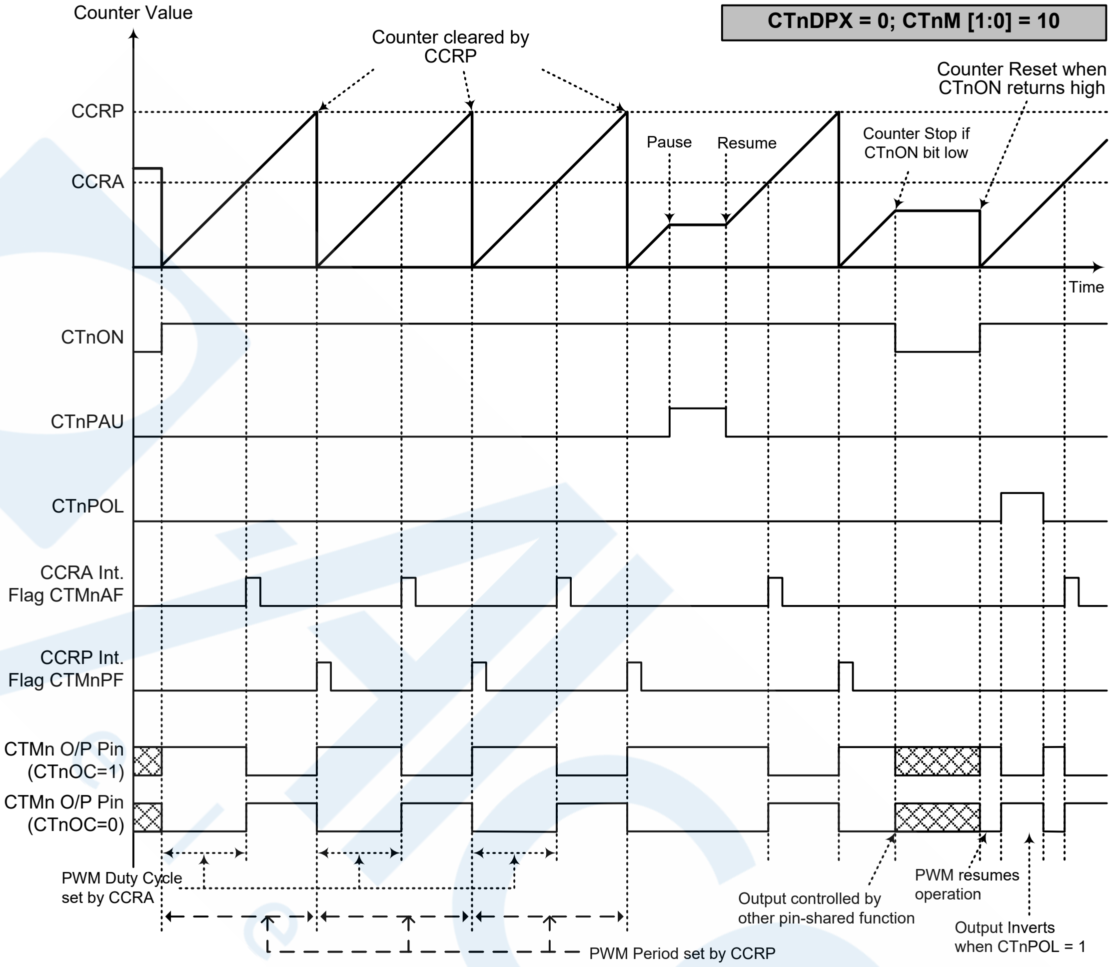
The CTMn PWM output frequency=$(f_{SYS}/4)/512=f_{SYS}/2048$=3.9063kHz, duty=128/512=25%.

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

- **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.



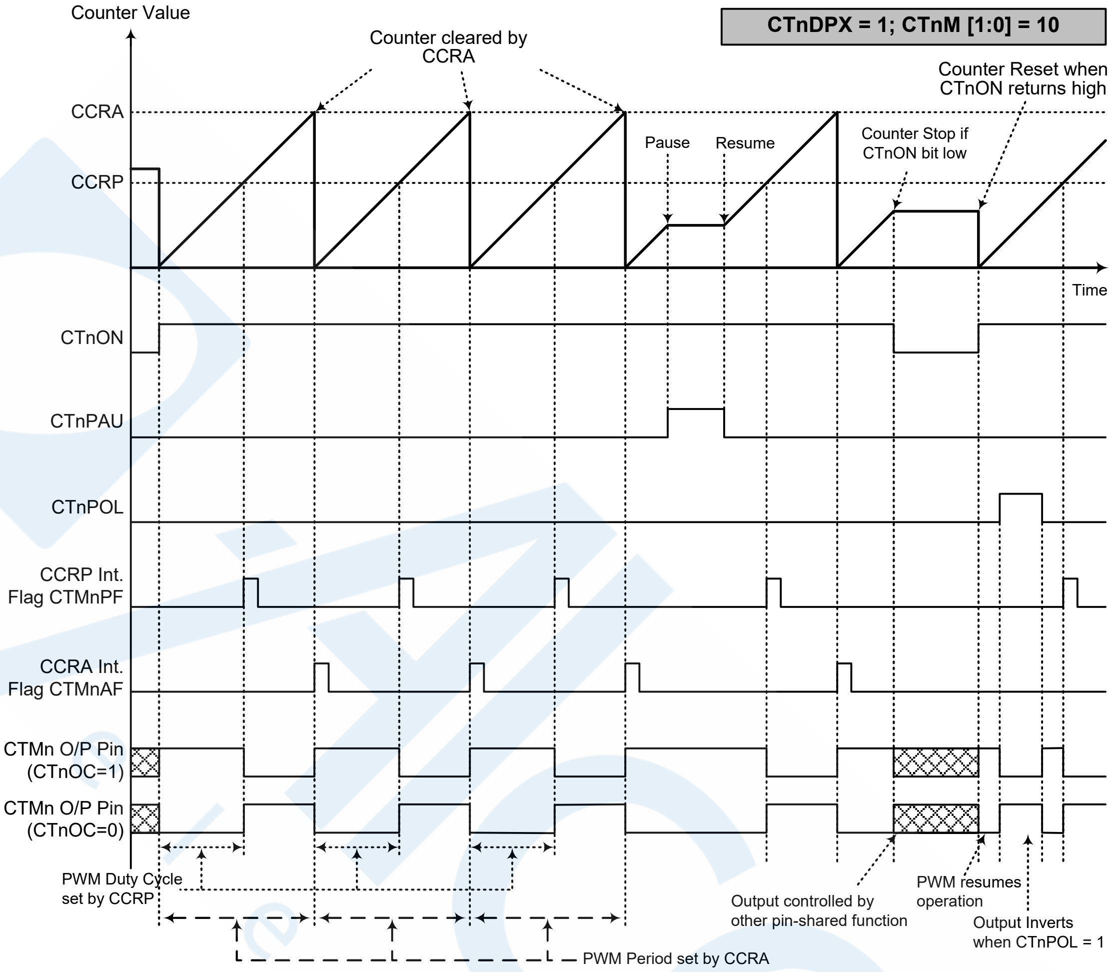
PWM Output Mode – CTnDPX=0 (n=0~1)

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

2. A counter clear sets PWM Period

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

4. The CTnCCLR bit has no influence on PWM operation



PWM Output Mode – CTnDPX=1 (n=0~1)

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

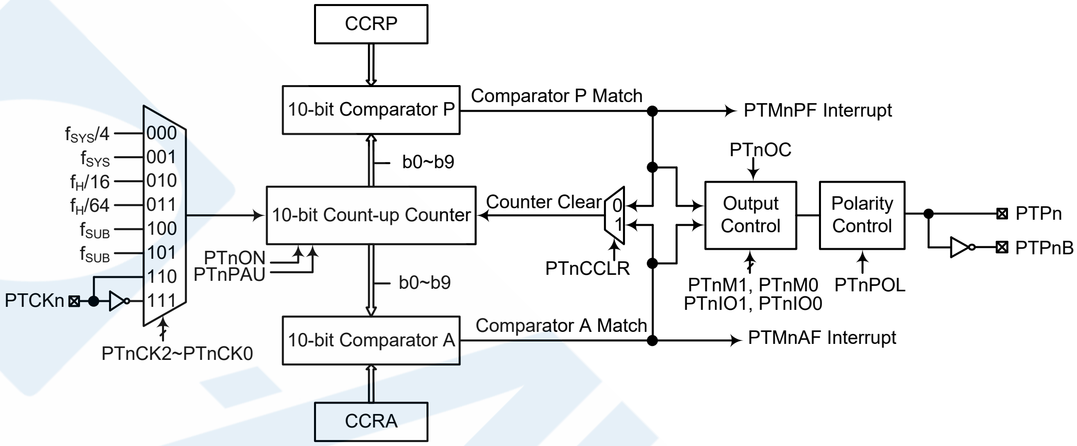
2. A counter clear sets PWM Period

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

4. The CTnCCLR bit has no influence on PWM operation

Periodic Type TM – PTM

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



Note: 1. The PTMn external pins are pin-shared with other functions, so before using the PTMn function, ensure that the pin-shared function registers have be set properly to enable the PTMn pin function. The PTCKn pin, if used, must also be set as an input by setting the corresponding bits in the port control register.
2. The PTPnB is the inverted signal of the PTPn.

10-bit Periodic Type TM Block Diagram (n=0)

Periodic Type TM Operation

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

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

Periodic Type TM Register Description

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

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

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

- **PTMnC0 Register**

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

Bit 7 **PTnPAU**: PTMn counter pause control
0: Run
1: Pause

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

Bit 6~4 **PTnCK2~PTnCK0**: PTMn counter clock selection
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: PTCKn rising edge clock
111: PTCKn falling edge clock

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

Bit 3 **PTnON**: PTMn counter on/off control
0: Off
1: On

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

If the PTMn is in the Compare Match Output Mode or PWM output Mode or Single Pulse Output Mode, then the PTMn output pin will be reset to its initial condition, as specified by the PTnOC bit, when the PTnON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

- **PTMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	D1	PTnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PTnM1~PTnM0**: PTMn operating mode selection
00: Compare Match Output Mode
01: Undefined
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

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

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

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

Timer/Counter Mode
Unused

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

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

In the PWM Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn output pin changes state when a certain compare match condition occurs. The PTMn output function is modified by changing these two bits. It is necessary to change the values of the PTnIO1 and PTnIO0 bits only after the PTMn has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the PTMn is running.

Bit 3 **PTnOC**: PTMn PTPn output control

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

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

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

Bit 2 **PTnPOL**: PTMn PTPn output polarity control
0: Non-invert
1: Invert

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

Bit 1 **D1**: Reserved bit, must be fixed at “0”

Bit 0 **PTnCCLR**: PTMn 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 Periodic TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the PTnCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The PTnCCLR bit is not used in the PWM Output or Single Pulse Output Mode.

- **PTMnDL Register**

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

Bit 7~0 **D7~D0**: PTMn Counter Low Byte Register bit 7 ~ bit 0
PTMn 10-bit Counter bit 7 ~ bit 0

- **PTMnDH Register**

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

Bit 7~2 Unimplemented, read as “0”

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

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

- **PTMnAH Register**

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

Bit 7~2	Unimplemented, read as “0”
Bit 1~0	**D9~D8**: PTMn CCRA High Byte Register bit 1 ~ bit 0 PTMn 10-bit CCRA bit 9 ~ bit 8

- **PTMnRPL Register**

Bit	7	6	5	4	3	2	1	0
Name	PTnRP7	PTnRP6	PTnRP5	PTnRP4	PTnRP3	PTnRP2	PTnRP1	PTnRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0	**PTnRP7~PTnRP0**: PTMn CCRP Low Byte Register bit 7 ~ bit 0 PTMn 10-bit CCRP bit 7 ~ bit 0

- **PTMnRPH Register**

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

Bit 7~2	Unimplemented, read as “0”
Bit 1~0	**PTnRP9~PTnRP8**: PTMn CCRP High Byte Register bit 1 ~ bit 0 PTMn 10-bit CCRP bit 9 ~ bit 8

Periodic Type TM Operation Modes

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

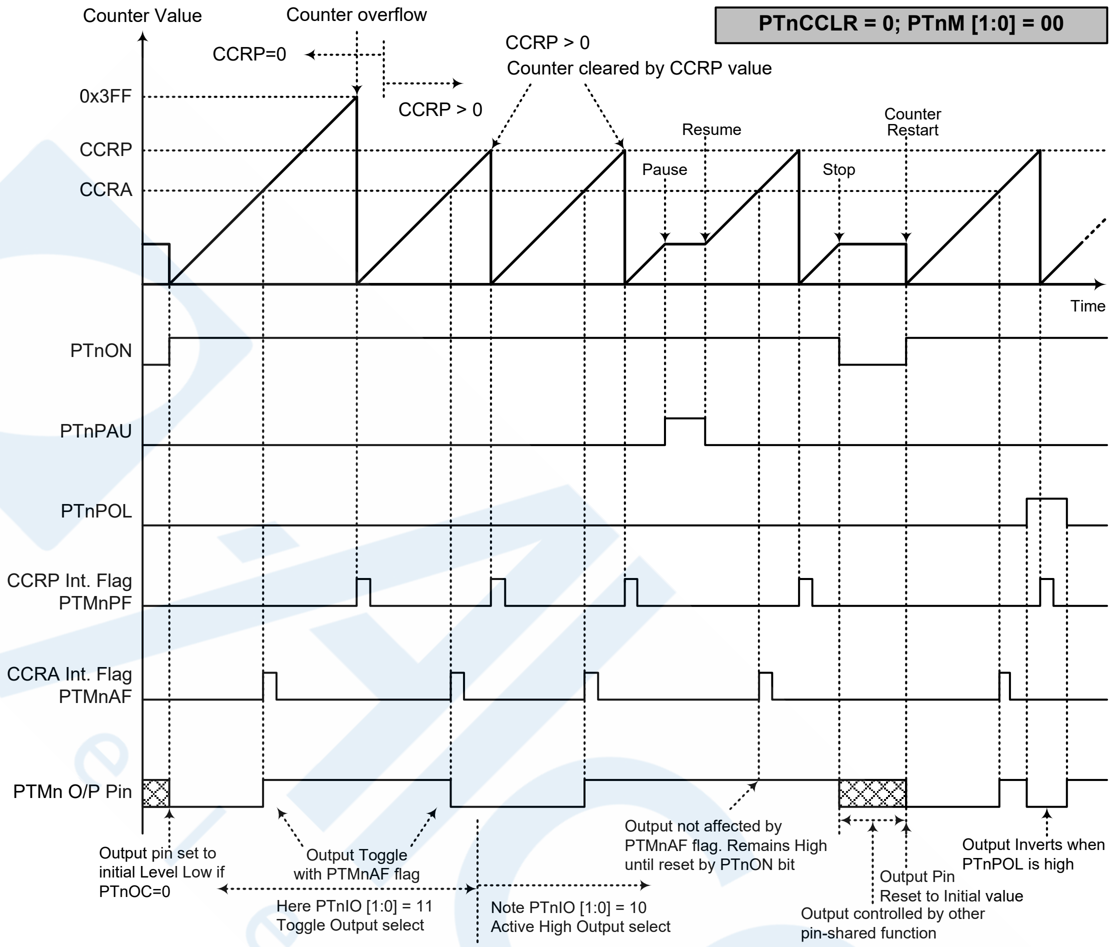
Compare Match Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register, should be set to “00” respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the PTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both PTMnAF and PTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the PTnCCLR bit in the PTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is high no PTMnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”.

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

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

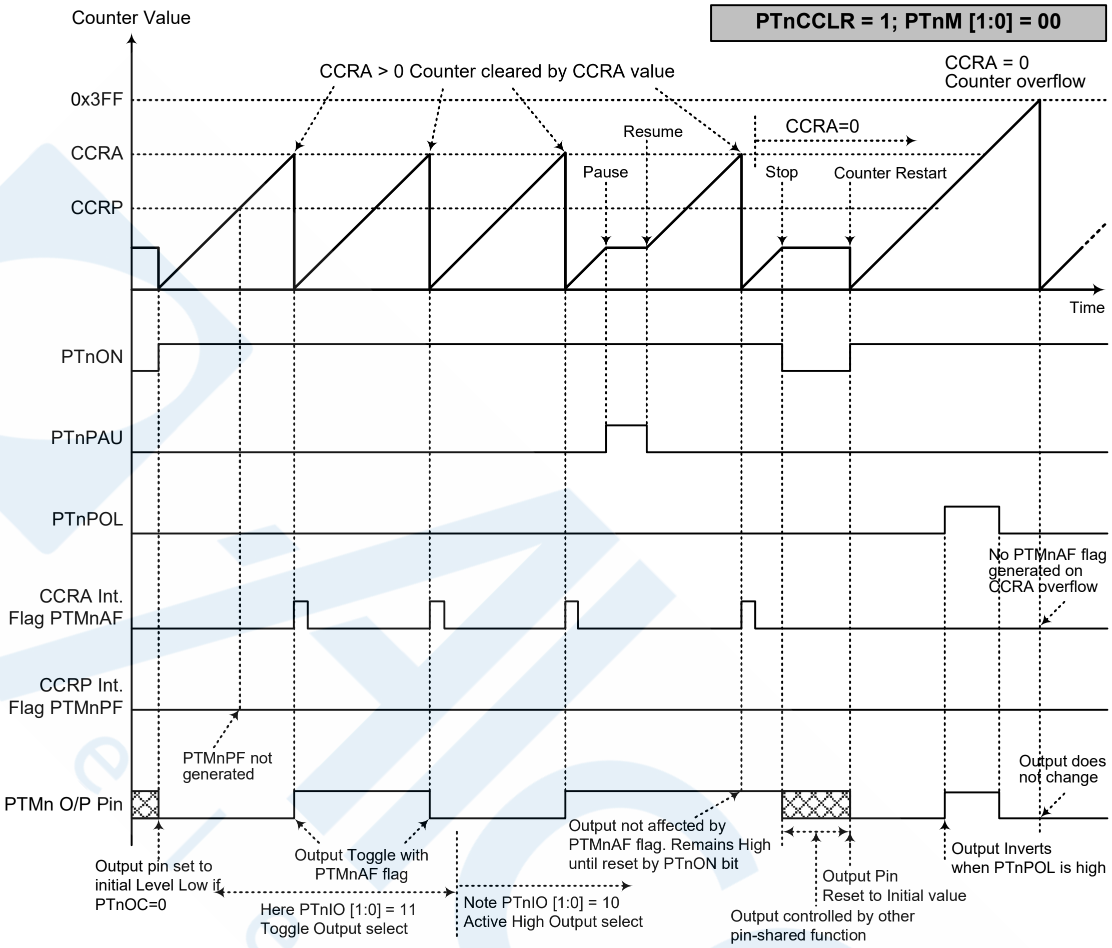


Compare Match Output Mode – PTnCCLR=0 (n=0)

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

2. The PTMn output pin is controlled only by the PTMnAF flag

3. The output pin is reset to its initial state by a PTnON bit rising edge



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

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

Timer/Counter Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to "11" respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the PTMn 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 PTMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared functions.

PWM Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to "10" respectively and also the PTnIO1 and PTnIO0 bits should be set to "10" respectively. The PWM function within the PTMn is useful for applications which require functions such as motor control, heating control, illumination control, etc. By providing a signal of fixed frequency but of varying duty cycle on the PTMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

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

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

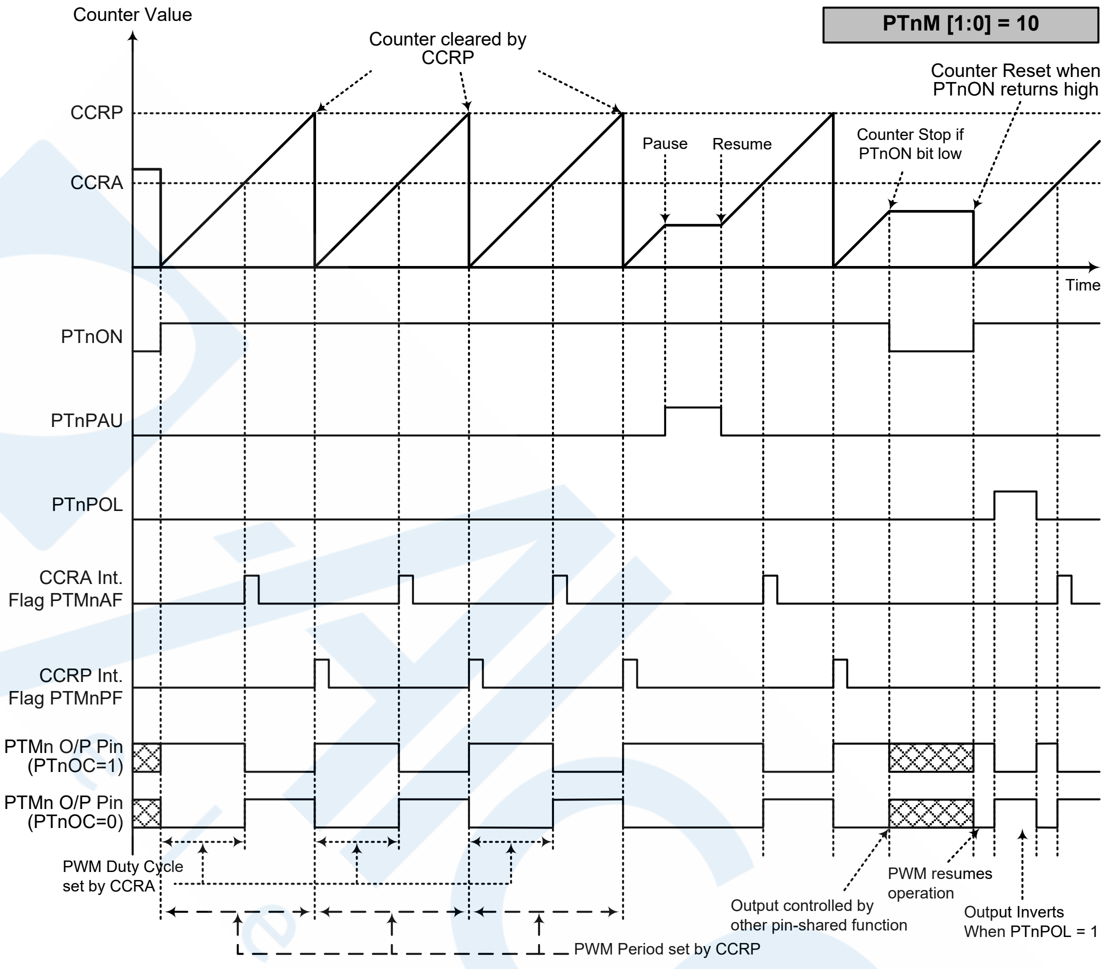
- **10-bit PWM Output Mode, Edge-aligned Mode**

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

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

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

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



PWM Output Mode (n=0)

Note: 1. The counter is cleared by CCRP

2. A counter clear sets the PWM Period

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

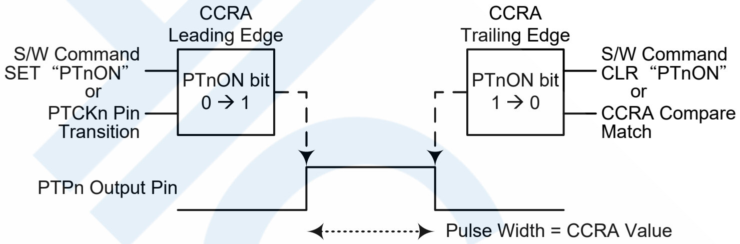
4. The PTnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

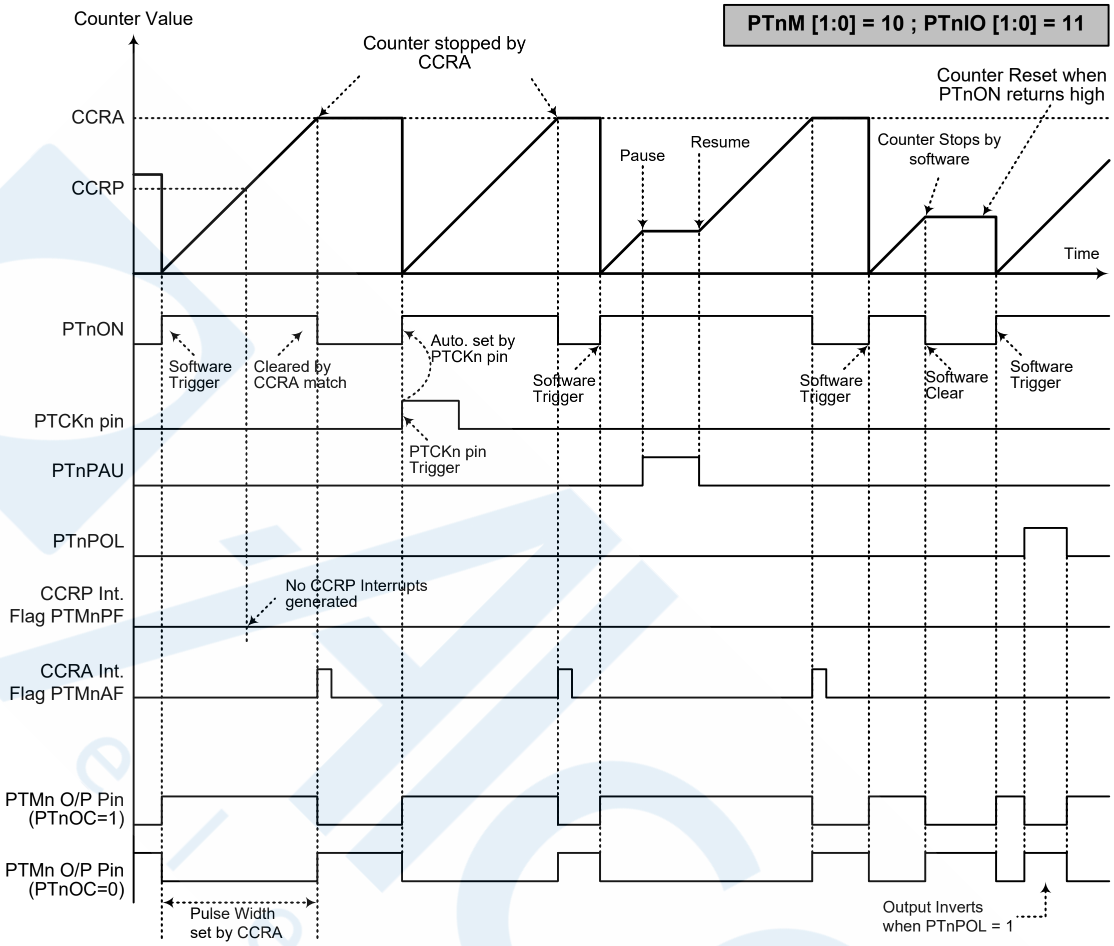
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to "10" respectively and also the PTnIO1 and PTnIO0 bits should be set to "11" respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

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

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



Single Pulse Generation (n=0)



Single Pulse Output Mode (n=0)

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

Touch Key Function

The device provides multiple touch key functions. The touch key function is fully integrated and requires no external components, allowing touch key functions to be implemented by the simple manipulation of internal registers.

Touch Key Structure

The touch keys are pin-shared with the I/O pins, with the desired function chosen via the corresponding selection register bits. Keys are organised into several groups, with each group known as a module and having a module number, M0 to Mn. Each module is a fully independent set of four Touch Keys and each key has its own oscillator. Each module contains its own control logic circuits and register set. Examination of the register names will reveal the module number it is referring to.

Total Key Number	Touch Key Module	Touch Key
16	M0	KEY1~KEY4
	M1	KEY5~KEY8
	M2	KEY9~KEY12
	M3	KEY13~KEY16

Touch Key Structure

Touch Key Register Definition

Each touch key module, which contains four touch key functions, is controlled using several registers. The following table shows the register set for each touch key module. The Mn within the register name refers to the touch key module number.

Register Name	Description
TKTMR	Touch key time slot 8-bit counter preload register
TKC0	Touch key function control register 0
TKC1	Touch key function control register 1
TK16DL	Touch key function 16-bit counter low byte
TK16DH	Touch key function 16-bit counter high byte
TKMn16DL	Touch key module n 16-bit C/F counter low byte
TKMn16DH	Touch key module n 16-bit C/F counter high byte
TKMnROL	Touch key module n reference oscillator capacitor selection low byte
TKMnROH	Touch key module n reference oscillator capacitor selection high byte
TKMnC0	Touch key module n control register 0
TKMnC1	Touch key module n control register 1

Touch Key Function Register Definition (n=0~3)

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	—	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0
TKC1	—	—	—	—	—	—	TKFS1	TKFS0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKMn16DL	D7	D6	D5	D4	D3	D2	D1	D0
TKMn16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKMnROL	D7	D6	D5	D4	D3	D2	D1	D0
TKMnROH	—	—	—	—	—	—	D9	D8

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKMnC0	MnMXS1	MnMXS0	MnDFEN	MnFILEN	MnSOFC	MnSOF2	MnSOF1	MnSOF0
TKMnC1	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN

Touch Key Function Register List (n=0~3)

- **TKTMR 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**: Touch key time slot 8-bit counter preload register

The touch key time slot counter preload register is used to determine the touch key time slot overflow time. The time slot unit period is obtained by a 5-bit counter and is equal to 32 time slot clock cycles. Therefore, the time slot counter overflow time is equal to the following equation shown.

Time slot counter overflow time = (256-TKTMR[7:0])×32 t_{TSC}, where t_{TSC} is the time slot counter clock period.

- **TKC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0
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 **TKRCOV**: Touch key time slot counter overflow flag
0: No overflow occurs
1: Overflow occurs

When this bit is set by touch key time slot counter overflow, the corresponding touch key interrupt request flag will be set. However, if this bit is set by application program, the touch key interrupt request flag will not be affected. Therefore, this bit cannot be set by application program but must be cleared to 0 by application program.

If the module 0 or all module time slot counter, selected by the TSCS bit, overflows, the TKRCOV bit and the Touch Key Interrupt request flag, TKMF, will be set and all module key oscillators and reference oscillators will automatically stop. The touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off.

Bit 5 **TKST**: Touch key detection start control
0: Stopped or no operation
0→1: Start detection

In all modules the touch key module 16-bit C/F counter, touch key function 16-bit counter and 5-bit time slot unit period counter will automatically be cleared when this bit is cleared to zero. However, the 8-bit programmable time slot counter will not be cleared. When this bit is changed from low to high, the touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be switched on together with the key and reference oscillators to drive the corresponding counters.

Bit 4 **TKCFOV**: Touch key module 16-bit C/F counter overflow flag
0: No overflow occurs
1: Overflow occurs

This bit is set high by the touch key module 16-bit C/F counter overflow and must be cleared to 0 by application programs.

Bit 3	**TK16OV**: Touch key function 16-bit counter overflow flag 0: No overflow occurs 1: Overflow occurs This bit is set high by the touch key function 16-bit counter overflow and must be cleared to 0 by application programs.
Bit 2	**TSCS**: Touch key time slot counter select 0: Each touch key module uses its own time slot counter 1: All touch key modules use Module 0 time slot counter
Bit 1~0	**TK16S1~TK16S0**: Touch key function 16-bit counter clock source select 00: f_{SYS} 01: $f_{SYS}/2$ 10: $f_{SYS}/4$ 11: $f_{SYS}/8$

• TKC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TKFS1	TKFS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	1	1

Bit 7~2	Unimplemented, read as “0”
Bit 1~0	**TKFS1~TKFS0**: Touch key oscillator and Reference oscillator frequency select 00: 1MHz 01: 3MHz 10: 7MHz 11: 11MHz

• TK16DH/TK16DL – Touch Key Function 16-bit Counter Register Pair

Register	TK16DH								TK16DL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register pair is used to store the touch key function 16-bit counter value. This 16-bit counter can be used to calibrate the reference or key oscillator frequency. When the touch key time slot counter overflows, this 16-bit counter will be stopped and the counter content will be unchanged. This register pair will be cleared to zero when the TKST bit is cleared to zero.

• TKMn16DH/TKMn16DL – Touch Key Module n 16-bit C/F Counter Register Pair

Register	TKMn16DH								TKMn16DL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register pair is used to store the touch key module n 16-bit C/F counter value. This 16-bit C/F counter will be stopped and the counter content will be kept unchanged when the touch key time slot counter overflows. This register pair will be cleared to zero when the TKST bit is cleared to zero.

- **TKMnROH/TKMnROL – Touch Key Module n Reference Oscillator Capacitor Selection Register Pair**

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

This register pair is used to store the touch key module n reference oscillator capacitor value.

The reference oscillator internal capacitor value = (TKMnRO[9:0]×50pF)/1024

- **TKMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	MnMXS1	MnMXS0	MnDFEN	MnFILEN	MnSOFC	MnSOF2	MnSOF1	MnSOF0
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 **MnMXS1~MnMXS0**: Multiplexer Key Selection

Bit	Touch Key Module Number			
MnMXS[1:0]	M0	M1	M2	M3
00	KEY1	KEY5	KEY9	KEY13
01	KEY2	KEY6	KEY10	KEY14
10	KEY3	KEY7	KEY11	KEY15
11	KEY4	KEY8	KEY12	KEY16

Bit 5 **MnDFEN**: Touch key module n multi-frequency control
0: Disable
1: Enable

This bit is used to control the touch key oscillator frequency doubling function. When this bit is set to 1, the key oscillator frequency will be doubled.

Bit 4 **MnFILEN**: Touch key module n filter function control
0: Disable
1: Enable

Bit 3 **MnSOFC**: Touch key module n C-to-F oscillator frequency hopping function control selection
0: Controlled by the MnSOF2~MnSOF0
1: Controlled by hardware circuit

This bit is used to select the touch key oscillator frequency hopping function control method. When this bit is set to 1, the key oscillator frequency hopping function is controlled by the hardware circuit regardless of the MnSOF2~MnSOF0 bits value.

Bit 2~0 **MnSOF2~MnSOF0**: Touch key module n Reference and Key oscillators hopping frequency selection (when MnSOFC=0)
000: 1.020MHz
001: 1.040MHz
010: 1.059MHz
011: 1.074MHz
100: 1.085MHz
101: 1.099MHz
110: 1.111MHz
111: 1.125MHz

These bits are used to select the touch key oscillator frequency for the hopping function. Note that these bits are only available when the MnSOFC bit is cleared to 0.

The frequencies mentioned here are only for the condition where the key and reference oscillator frequency is selected to be 1MHz, these values will be changed when the external or internal capacitor has different values. Users can adjust the key and reference oscillator frequency in scale when any other frequency is selected.

- **TKMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN
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 **MnTSS**: Touch key module n time slot counter clock source Selection
0: Touch key module n reference oscillator
1: $f_{SYS}/4$

Bit 6 Unimplemented, read as "0"

Bit 5 **MnROEN**: Touch key module n Reference oscillator enable control
0: Disable
1: Enable

Bit 4 **MnKOEN**: Touch key module n Key oscillator enable control
0: Disable
1: Enable

Bit 3 **MnK4EN**: Touch key module n Key 4 enable control

MnK4EN	Touch Key Module n (Mn)			
	M0	M1	M2	M3
0: Disable	I/O or other functions			
1: Enable	KEY4	KEY8	KEY12	KEY16

Bit 2 **MnK3EN**: Touch key module n Key 3 enable control

MnK3EN	Touch Key Module n (Mn)			
	M0	M1	M2	M3
0: Disable	I/O or other functions			
1: Enable	KEY3	KEY7	KEY11	KEY15

Bit 1 **MnK2EN**: Touch key module n Key 2 enable control

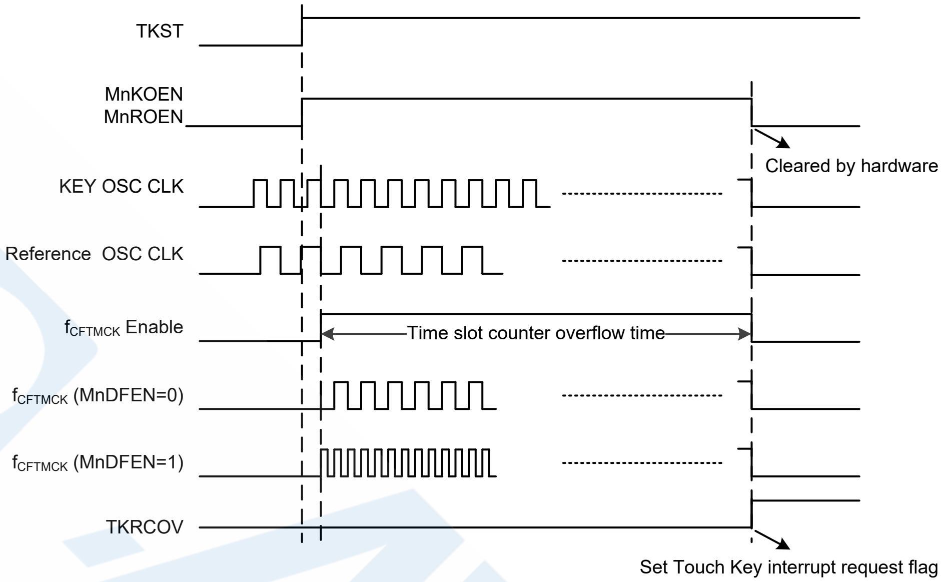
MnK2EN	Touch Key Module n (Mn)			
	M0	M1	M2	M3
0: Disable	I/O or other functions			
1: Enable	KEY2	KEY6	KEY10	KEY14

Bit 0 **MnK1EN**: Touch key module n Key 1 enable control

MnK1EN	Touch Key Module n (Mn)			
	M0	M1	M2	M3
0: Disable	I/O or other functions			
1: Enable	KEY1	KEY5	KEY9	KEY13

Touch Key Operation

When a finger touches or is in proximity to a touch pad, the capacitance of the pad will increase. By using this capacitance variation to change slightly the frequency of the internal sense oscillator, touch actions can be sensed by measuring these frequency changes. Using an internal programmable divider the reference clock is used to generate a fixed time period. By counting a number of generated clock cycles from the sense oscillator during this fixed time period touch key actions can be determined.



Touch Key Timing Diagram

Each touch key module contains four touch key inputs which are shared with logical I/O pins, and the desired function is selected using the relevant pin-shared control register bits. Each touch key has its own independent sense oscillator. Therefore, there are four sense oscillators within each touch key module.

During this reference clock fixed interval, the number of clock cycles generated by the sense oscillator is measured, and it is this value that is used to determine if a touch action has been made or not. At the end of the fixed reference clock time interval, a Touch Key interrupt signal will be generated.

Using the TSCS bit in the TKC0 register can select the module 0 time slot counter as the time slot counter for all modules. All modules use the same started signal, TKST, in the TKC0 register. The touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter in all modules will be automatically cleared when the TKST bit is cleared to zero, but the 8-bit programmable time slot counter will not be cleared. The overflow time is setup by user. When theTKST bit changes from low to high, the 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched on.

The key oscillator and reference oscillator in all modules will be automatically stopped and the 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched off when the time slot counter overflows. The clock source for the time slot counter is sourced from the reference oscillator or $f_{SYS}/4$ which is selected using the MnTSS bit in the TKMnC1 register. The reference oscillator and key oscillator will be enabled by setting the MnROEN bit and MnKOEN bits in the TKMnC1 register.

When the time slot counter in all the touch key modules or in the touch key module 0 overflows, an actual touch key interrupt will take place. The touch keys mentioned here are the keys which are enabled.

Each touch key module consists of four touch keys, KEY1~KEY4 are contained in module 0, KEY5~KEY8 are contained in module 1, KEY9~KEY12 are contained in module 2, etc. Each touch key module has an identical structure.

Touch Key Interrupt

The touch key only has single interrupt, when the time slot counter in all the touch key modules or in the touch key module 0 overflows, an actual touch key interrupt will take place. The touch keys mentioned here are the keys which are enabled. The 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter in all modules will be automatically cleared. More details regarding the touch key interrupt is located in the interrupt section of the datasheet.

Programming Considerations

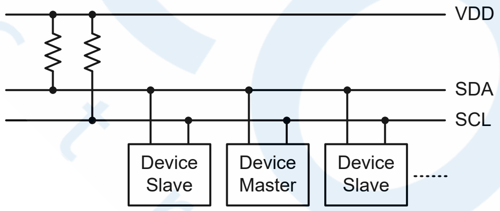
After the relevant registers are setup, the touch key detection process is initiated by changing the TKST bit from low to high. This will enable and synchronise all relevant oscillators. The TKRCOV flag which is the time slot counter flag will go high when the counter overflows. When this happens an interrupt signal will be generated. As the TKRCOV flag will not be automatically cleared, it has to be cleared by the application program.

The TKCFOV flag which is the 16-bit C/F counter overflow flag will go high when any of the Touch Key Module 16-bit C/F counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program. The TK16OV flag which is the 16-bit counter overflow flag will go high when the 16-bit counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program.

When the external touch key size and layout are defined, their related capacitances will then determine the sensor oscillator frequency.

I^{2}C Interface

The I^{2}C 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.

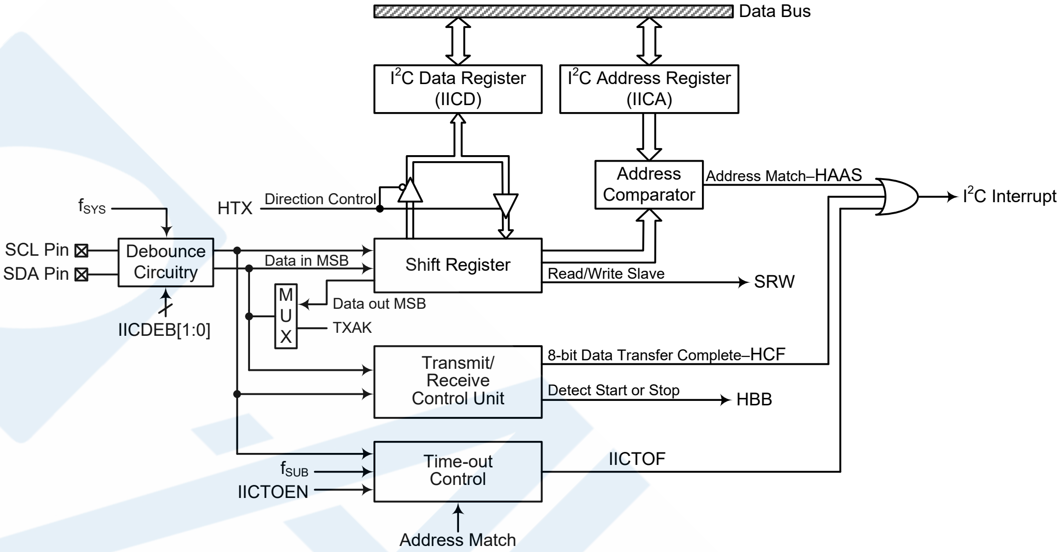


I^{2}C Master Slave Bus Connection

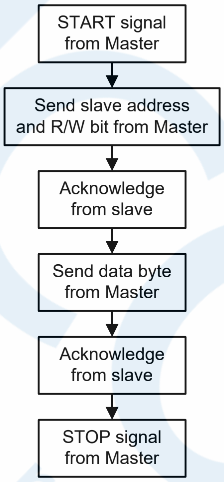
I^{2}C Interface Operation

The I^{2}C 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 I^{2}C bus is identified by a unique address which will be transmitted and received on the I^{2}C bus.

When two devices communicate with each other on the bidirectional I^{2}C 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 I^{2}C 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 function could be controlled by its corresponding pull-high control register. It is suggested that the device should not enter the IDLE/SLEEP mode during the I²C communication.



I²C Block Diagram



I²C Interface Operation

The IICDEB1 and IICDEB0 bits determine the debounce time of the I²C interface. This uses the internal clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required I²C data transfer speed, there exists a relationship between the system clock, f_{SYS}, and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

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

I²C Minimum f_{SYS} Frequency Requirements

I²C Registers

There are three control registers associated with the I²C bus, IICC0, IICC1 and IICTOC, one address register IICA and one data register, IICD.

Register Name	Bit							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
IICC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
IICD	D7	D6	D5	D4	D3	D2	D1	D0
IICA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
IICTOC	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0

I²C Register List

I²C Data Register

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

- **IICD 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**: I²C data register bit 7 ~ bit 0

I²C Address Register

The IICA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the IICA 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 IICA register, the slave device will be selected.

- **IICA Register**

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

Bit 0 Unimplemented, read as "0"

I²C Control Registers

There are three control registers for the I²C interface, IICC0, IICC1 and IICTOC. The IICC0 register is used to control the enable/disable function and to set the debounce time. The IICC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, IICTOC, is used to control the I²C time-out function and is described in the corresponding section.

- **IICC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **IICDEB1~IICDEB0**: I^{2}C debounce time selection
00: No debounce
01: 2 system clock debounce
1x: 4 system clock debounce

Note that the I^{2}C debounce circuit will operate normally if the system clock, f_{SYS}, is derived from the f_H clock or the IAMWU bit is equal to 0. Otherwise, the debounce circuit will have no effect and be bypassed.

Bit 1 **ICCEN**: I^{2}C enable control
0: Disable
1: Enable

The bit is the overall on/off control for the I^{2}C interface. When the ICCEN bit is cleared to zero to disable the I^{2}C interface, the SDA and SCL lines will lose their I^{2}C function and the I^{2}C operating current will be reduced to a minimum value. When the bit is high the I^{2}C interface is enabled. If the IICEN bit changes from low to high, the contents of the I^{2}C 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^{2}C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 Unimplemented, read as "0"

- **IICC1 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**: I^{2}C 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. Below is an example of the flow of a two-byte I^{2}C data transfer. First, I^{2}C slave device receives a start signal from I^{2}C master and then HCF bit is automatically cleared to zero. Second, I^{2}C slave device finishes receiving the 1st data byte and then HCF bit is automatically set high. Third, user read the 1st data byte from IICD register by the application program and then HCF bit is automatically cleared to zero. Fourth, I^{2}C slave device finishes receiving the 2nd data byte and then HCF bit is automatically set to one and so on. Finally, I^{2}C slave device receives a stop signal from I^{2}C master and then HCF bit is automatically set high.

Bit 6 **HAAS**: I^{2}C 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**: I^{2}C bus busy flag
0: I^{2}C Bus is not busy
1: I^{2}C Bus is busy

The HBB flag is the I^{2}C busy flag. This flag will be "1" when the I^{2}C 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 match, which is when the HAAS flag is set high, the slave device will check the SRW flag to determine whether it should be in transmit mode or receive mode. If the SRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

Bit 1 **IAMWU**: I²C address match wake-up control
0: Disable
1: Enable

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

Bit 0 **RXAK**: I²C bus receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave does not receive acknowledge flag

The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

I²C Bus Communication

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the IICC1 register will be set and an I²C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and IICTOF 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

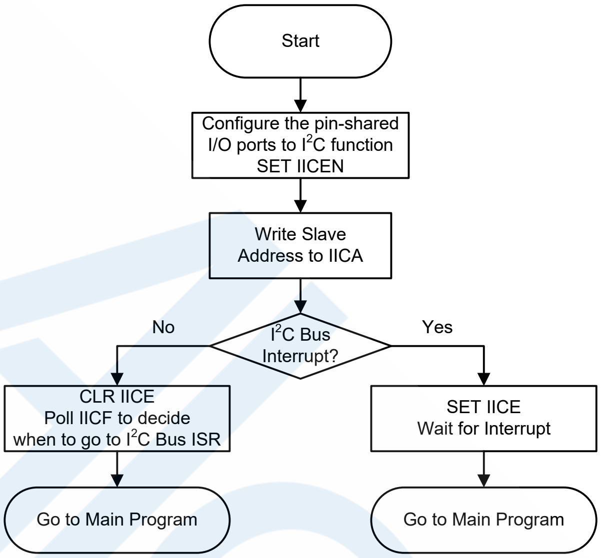
  Configure the corresponding pin-shared function as the I²C function pins and set the IICEN bit to "1" to enable the I²C bus.

- Step 2

  Write the slave address of the device to the I²C bus address register IICA.

- Step 3

  Set the I²C interrupt enable bit of the interrupt control register to enable the I²C 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 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 IICC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

As an I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and IICTOF 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 IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the SCL line.

I^{2}C Bus Read/Write Signal

The SRW bit in the IICC1 register defines whether the master device wishes to read data from the I^{2}C bus or write data to the I^{2}C 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 I^{2}C bus, therefore the slave device must be setup to send data to the I^{2}C bus as a transmitter. If the SRW flag is "0" then this indicates that the master wishes to send data to the I^{2}C bus, therefore the slave device must be setup to read data from the I^{2}C bus as a receiver.

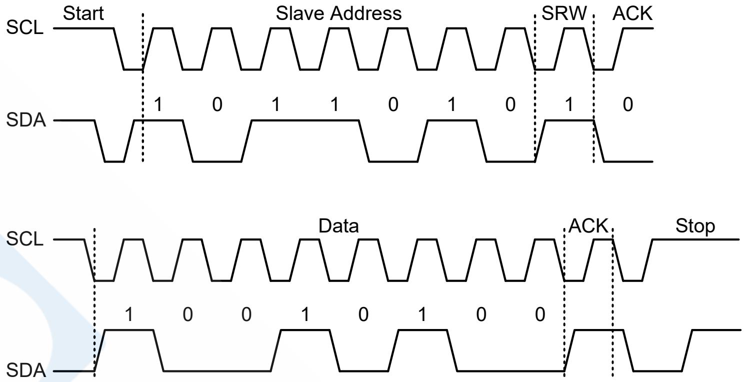
I^{2}C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I^{2}C 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 IICC1 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 IICC1 register should be set to "0".

I^{2}C 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 I^{2}C Bus. The corresponding data will be stored in the IICD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the IICD register. If setup as a receiver, the slave device must read the transmitted data from the IICD 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 IICC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.

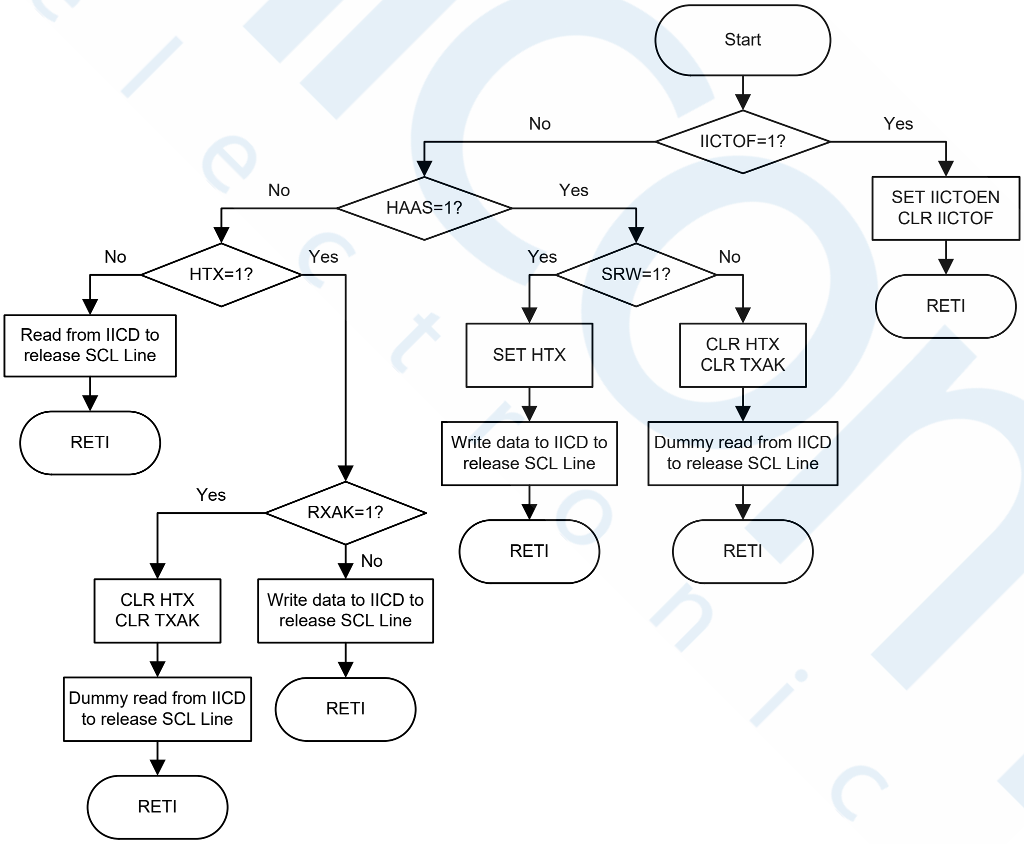


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	……	S	SA	SR	M	D	A	D	A	……	P

I²C Communication Timing Diagram

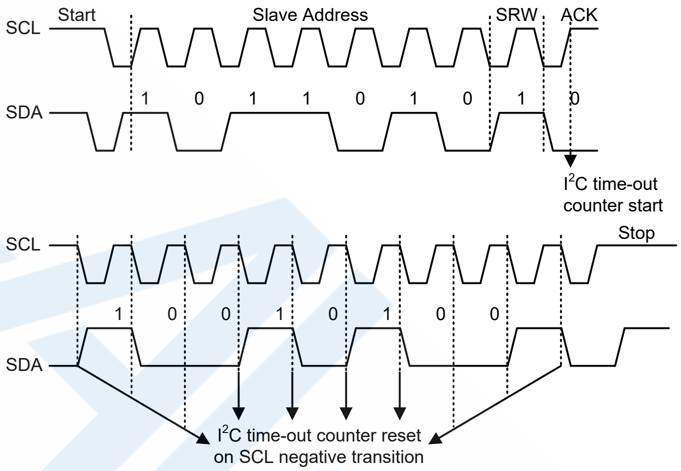
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the SCL line.



I²C Bus ISR Flow Chart

I^{2}C Time-out Control

In order to reduce the problem of I^{2}C lockup due to reception of erroneous clock sources, a time-out function is provided. If the clock source to the I^{2}C is not received for a while, then the I^{2}C circuitry and registers will be reset after a certain time-out period. The time-out counter starts counting on an I^{2}C 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 IICTOC register, then a time-out condition will occur. The time-out function will stop when an I^{2}C “STOP” condition occurs.



I^{2}C Time-out

When an I^{2}C time-out counter overflow occurs, the counter will stop and the IICTOEN bit will be cleared to zero and the IICTOF 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 I^{2}C interrupt vector. When an I^{2}C time-out occurs, the I^{2}C internal circuitry will be reset and the registers will be reset into the following condition:

Registers	After I^{2}C Time-out
IICD, IICA, IICC0	No change
IICC1	Reset to POR condition

I^{2}C Registers after Time-out

The IICTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using IICTOS5~IICTOS0 bits in the IICTOC register. The time-out time is given by the formula: $((1\sim64)\times(32/f_{SUB}))$. This gives a time-out period which ranges from about 1ms to 64ms.

- **IICTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0
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 **IICTOEN**: I^{2}C time-out control
0: Disable
1: Enable

Bit 6 **IICTOF**: I^{2}C time-out flag
0: No time-out occurred
1: Time-out occurred
This bit is set high when time-out occurs and can only be cleared by application program.

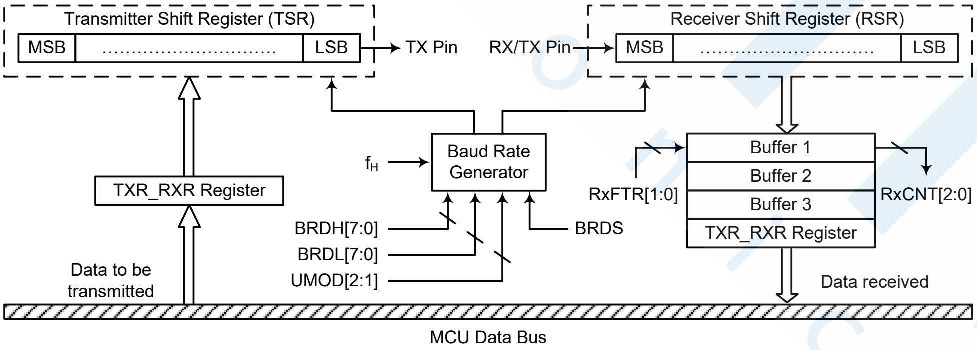
Bit 5~0 **IICTOS5~IICTOS0**: I^2C time-out period selection
I^2C time-out clock source is $f_{SUB}/32$.
I^2C time-out time is equal to $(IICTOS[5:0]+1)\times(32/f_{SUB})$.

UART Interface

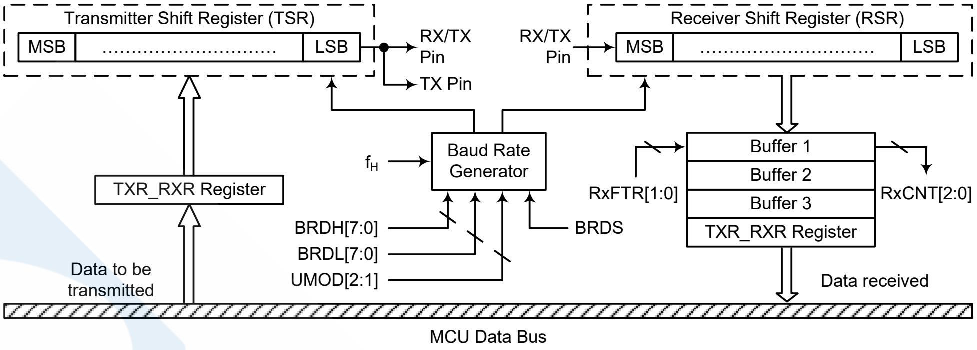
The device contains an integrated full-duplex or half-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data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

The integrated UART function contains the following features:

- Full-duplex or half-duplex (single wire mode) asynchronous communication
- 8 or 9 bits character length
- Even, odd, mark, space or no parity options
- One or two stop bits configurable for receiver
- Two stop bits for transmitter
- Baud rate generator with 16-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 4-byte Deep FIFO Receive Data Buffer
- 1-byte Deep FIFO Transmit Data Buffer
- RX/TX pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
  - Transmitter Empty
  - Transmitter Idle
  - Receiver Full
  - Receiver Overrun
  - Address Mode Detect



UART Data Transfer Block Diagram – SWM=0



UART Data Transfer Block Diagram – SWM=1

UART External Pins

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

UART Single Wire Mode

The UART function also supports the Single Wire Mode communication which is selected using the SWM bit in the UCR3 register. When the SWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single RX/TX pin can be used to transmit and receive data depending upon the corresponding control bits. When the RXEN bit is set high, the RX/TX pin is used as a receiver pin. When the RXEN bit is cleared to zero and the TXEN bit is set high, the RX/TX pin will act as a transmitter pin.

It is recommended not to set both the RXEN and TXEN bits high in the single wire mode. If both the RXEN and TXEN bits are set high, the RXEN bit will have the priority and the UART will act as a receiver.

It is important to note that the functional description in this UART Interface chapter, which is described from the full-duplex communication standpoint, also applies to the half-duplex (single wire mode) communication except the pin usage. In the single wire mode, the TX pin mentioned in this chapter should be replaced by the RX/TX pin to understand the whole UART single wire mode function.

In the single wire mode, the data can also be transmitted on the TX pin in a transmission operation with proper software configurations. Therefore, the data will be output on the RX/TX and TX pins.

UART Data Transfer Scheme

The UART Data Transfer Block Diagram shows the overall data transfer structure arrangement for the UART interface. The actual data to be transmitted from the MCU is first transferred to the TXR_RXR register by the application program. The data will then be transferred to the Transmit

Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate Generator. Only the TXR_RXR register is mapped onto the MCU Data Memory, the Transmit Shift Register is not mapped and is therefore inaccessible to the application program.

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

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

UART Status and Control Registers

There are nine control registers associated with the UART function. The SWM bit in the UCR3 register is used to enable/disable the UART Single Wire Mode. The USR, UCR1, UCR2, UFCR and RxCNT registers control the overall function of the UART, while the BRDH and BRDL registers control the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXR data register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
UCR3	—	—	—	—	—	—	—	SWM
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRDH	D7	D6	D5	D4	D3	D2	D1	D0
BRDL	D7	D6	D5	D4	D3	D2	D1	D0
UFCR	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
RxCNT	—	—	—	—	—	D2	D1	D0

UART Register List

• USR Register

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

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

Bit 7 **PERR**: Parity error flag
0: No parity error is detected
1: Parity error is detected

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

Bit 6 **NF**: Noise flag
0: No noise is detected
1: Noise is detected

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

Bit 5 **FERR**: Framing error flag
0: No framing error is detected
1: Framing error is detected

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

Bit 4 **OERR**: Overrun error flag
0: No overrun error is detected
1: Overrun error is detected

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

Bit 3 **RIDLE**: Receiver status
0: Data reception is in progress (Data being received)
1: No data reception is in progress (Receiver is idle)

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

Bit 2 **RXIF**: Receive TXR_RXR data register status
0: TXR_RXR data register is empty
1: TXR_RXR data register has available data and reaches receiver FIFO trigger level

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

Bit 1 **TIDLE**: Transmission status
0: Data transmission is in progress (Data being transmitted)
1: No data transmission is in progress (Transmitter is idle)

The TIDLE flag is known as the transmission complete flag. When this read only flag is "0", it indicates that a transmission is in progress. This flag will be set high when the TXIF flag is "1" and when there is no transmit data or break character being transmitted. When TIDLE is equal to "1", the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared by reading the USR register with

TIDLE set and then writing to the TXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 **TXIF**: Transmit TXR_RXR data register status
0: Character is not transferred to the transmit shift register
1: Character has transferred to the transmit shift register (TXR_RXR data register is empty)

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

• UCR1 Register

The UCR1 register together with the UCR2 and UCR3 registers are the three 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, single wire mode communication etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

“x”: unknown

Bit 7 **UARTEN**: UART function enable control
0: Disable UART. TX and RX/TX pins are in the floating state
1: Enable UART. TX and RX/TX pins function as UART pins

The UARTEN bit is the UART enable bit. When this bit is equal to “0”, the UART will be disabled and the RX/TX pin as well as the TX pin will be in the floating state. When the bit is equal to “1”, the UART will be enabled and the TX and RX/TX pins will function as defined by SWM mode selection bit together with the TXEN and RXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits as well as the RxCNT register will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

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

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

Note that the 9th bit of data if BNO=1, or the 8th bit of data if BNO=0, which is used as the parity bit, does not transfer to RX8 or TXRX7 respectively when the parity function is enabled.

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

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

Bit 4~3 **PRT1~PRT0**: Parity type selection bits
00: Even parity for parity generator
01: Odd parity for parity generator
10: Mark parity for parity generator
11: Space parity for parity generator

These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.

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

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

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

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

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

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

• UCR2 Register

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

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

Bit 7 **TXEN**: UART Transmitter enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled

The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to "0", the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TX pin will be in a floating state.

If the TXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be in a floating state.

Bit 6 **RXEN**: UART Receiver enabled control
0: UART receiver is disabled
1: UART receiver is enabled

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

Bit 5 **STOPS**: Number of stop bits selection for receiver
0: One stop bit format is used
1: Two stop bits format is used

This bit determines if one or two stop bits are to be used for receiver. 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. Two stop bits are used for transmitter.

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

The bit named ADDEN is the address detect function enable control bit. When this bit is equal to “1”, the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXRX7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of “1”, then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the received word is “0” with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **WAKE**: RX/TX pin wake-up UART function enable control
0: RX/TX pin wake-up UART function is disabled
1: RX/TX 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/TX pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no RX/TX pin wake-up UART function if the UART clock (f_H) exists. If the WAKE bit is set to 1 as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the RX/TX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX/TX 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/TX pin when the WAKE bit is cleared to 0.

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

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

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

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

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

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

- **UCR3 Register**

The UCR3 register is used to enable the UART Single Wire Mode communication. As the name suggests in the single wire mode the UART communication can be implemented in one single line, RX/TX, together with the control of the RXEN and TXEN bits in the UCR2 register.

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **SWM**: Single Wire Mode enable control
0: Disable, the RX/TX pin is used as UART receiver function only
1: Enable, the RX/TX pin can be used as UART receiver or transmitter function controlled by the RXEN and TXEN bits

Note that when the Single Wire Mode is enabled, if both the RXEN and TXEN bits are high, the RX/TX pin will only be used as UART receiver input.

- **TXR_RXR Register**

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

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

"x": unknown

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

- **BRDH 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**: Baud rate divider high byte

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

$$\text{Baud Rate} = f_H/(\text{BRD}+\text{UMOD}/8)$$

BRD = 16~65535 or 8~65535 depending on BRDS

Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.
2. The BRDL must be written first and then BRDH, otherwise errors may occur.
3. The BRDH register should not be modified during data transmission process.

• BRDL 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**: Baud rate divider low byte

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

$$\text{Baud Rate} = f_H/(\text{BRD}+\text{UMOD}/8)$$

BRD = 16~65535 or 8~65535 depending on BRDS

Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.
2. The BRDL must be written first and then BRDH, otherwise errors may occur.
3. The BRDL register should not be modified during data transmission process.

• UFCR Register

The UFCR register is the FIFO control register which is used for UART modulation control, BRD range selection and trigger level selection for RXIF and interrupt.

Bit	7	6	5	4	3	2	1	0
Name	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
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 5~3 **UMOD2~UMOD0**: UART Modulation Control bits

The modulation control bits are used to correct the baud rate of the received or transmitted UART signal. These bits determine if the extra UART clock cycle should be added in a UART bit time. The UMOD2~UMOD0 will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle.

Bit 2 **BRDS**: BRD range selection
0: BRD range is from 16 to 65535
1: BRD range is from 8 to 65535

The BRDS is used to control the sampling point in a UART bit time. If the BRDS is cleared to zero, the sampling point will be BRD/2, BRD/2+1×f_H, and BRD/2+2×f_H in a UART bit time. If the BRDS is set high, the sampling point will be BRD/2-1×f_H, BRD/2, and BRD/2+2×f_H in a UART bit time.

Note that the BRDS bit should not be modified during data transmission process.

Bit 1~0 **RxFTR1~RxFTR0**: Receiver FIFO trigger level (bytes)
00: 4 bytes in Receiver FIFO
01: 1 or more bytes in Receiver FIFO
10: 2 or more bytes in Receiver FIFO
11: 3 or more bytes in Receiver FIFO

For the receiver these bits define the number of received data bytes in the Receiver FIFO that will trigger the RXIF bit being set high, an interrupt will also be generated if the RIE bit is enabled. To prevent OERR from being set high, the receiver FIFO trigger level can be set to 2 bytes, avoiding an overrun state that cannot be processed by the program in time when more than 4 data bytes are received. After the reset the receiver FIFO is empty.

- **RxCNT Register**

The RxCNT register is the counter used to indicate the number of received data bytes in the Receiver FIFO which have not been read by the MCU. This register is read only.

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

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **D2~D0**: Receiver FIFO counter

The RxCNT register is the counter used to indicate the number of receiver data bytes in Receiver FIFO which is not read by MCU. When Receiver FIFO receives one byte data, the RxCNT will increase by one; when the MCU reads one byte data from Receiver FIFO, the RxCNT will decrease by one. If there are 4 bytes of data in the Receiver FIFO, the 5th data will be saved in the shift register. If there is 6th data, the 6th data will be saved in the shift register. But the RxCNT remains the value of 4. The RxCNT will be cleared when reset occurs or UARTEN=1. This register is read only.

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 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in BRDH/BRDL register and the second is the UART modulation control bits (UMOD2~UMOD0). To prevent accumulated error of the receiver baud rate frequency, it is recommended to use two stop bits for resynchronization after each byte is received. If a baud rate BR is required with UART clock $f_{H.}$

$$f_H/BR = \text{Integer Part} + \text{Fractional Part}$$

The integer part is loaded into BRD (BRDH/BRDL). The fractional part is multiplied by 8 and rounded, then loaded into UMOD bit field as following:

$$BRD = \text{TRUNC}\ (f_H/BR)$$

$$UMOD = \text{ROUND}\ [\text{MOD}\ (f_H/BR)\times 8]$$

Therefore, the actual baud rate is as following:

$$\text{Baud rate} = f_H/[BRD+(UMOD/8)]$$

Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, determine the BRDH/BRDL register value, the actual baud rate and the error value for a desired baud rate of 230400.

From the above formula, the BRD = TRUNC (f_H/BR) = TRUNC(17.36111) = 17

The UMOD = ROUND[MOD(f_H/BR)×8] = ROUND(0.36111×8) = ROUND(2.88888) = 3

The actual Baud Rate = f_H/[BRD+(UMOD/8)] = 230215.83

Therefore the error is equal to (230215.83-230400)/230400 = -0.08%

Modulation Control Example

To get the best-fitting bit sequence for UART modulation control bits UMOD2~UMOD0, the following algorithm can be used: Firstly, the fractional part of the theoretical division factor is multiplied by 8. Then the product will be rounded and UMOD2~UMOD0 bits will be filled with the rounded value. The UMOD2~UMOD0 bits will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle. The following is an example using the fraction 0.36111 previously calculated: UMOD[2:0]=ROUND(0.36111×8)=011b.

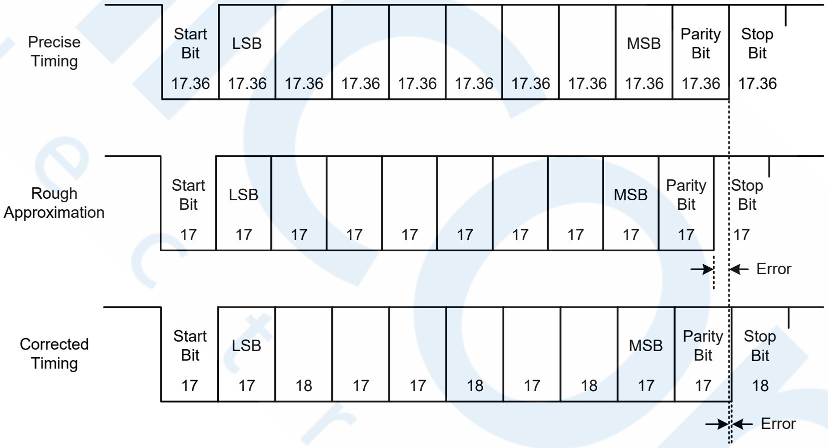
Fraction Addition	Carry to Bit 3	UART Bit Time Sequence	Extra UART Clock Cycle
0000b + 0011b=0011b	No	Start bit	No
0011b + 0011b=0110b	No	D0	No
0110b + 0011b=1001b	Yes	D1	Yes
1001b + 0011b=1100b	No	D2	No
1100b + 0011b=1111b	No	D3	No
1111b + 0011b=0010b	Yes	D4	Yes
0010b + 0011b=0101b	No	D5	No
0101b + 0011b=1000b	Yes	D6	Yes
1000b + 0011b=1011b	No	D7	No
1011b + 0011b=1110b	No	Parity bit	No
1110b + 0011b=0001b	Yes	Stop bit	Yes

Baud Rate Correction Example

The following figure presents an example using a baud rate of 230400 generated with UART clock f_H. The data format for the following figure is: eight data bits, parity enabled, no address bit; two stop bits.

The following figure shows three different frames:

- The upper frame is the correct one, with a bit-length of 17.36 f_H cycles (4000000/230400=17.36).
- The middle frame uses a rough estimate, with 17 f_H cycles for the bit length.
- The lower frame shows a corrected frame using the best fit for the UART modulation control bits UMOD2~UMOD0.



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 set to be even, odd, mark, space 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 along with the parity are setup by programming the BNO, PRT1~PRT0 and PREN bits. The transmitter always uses two stop bits while the receiver uses one or two stop bits which is determined by the STOPS bit. The baud rate used to transmit and receive data is set using the internal 16-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

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

Clearing the UARTEN bit will disable the TX and RX/TX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF as well as register RxCNT being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

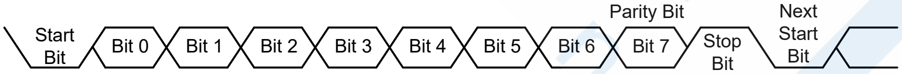
Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1and UCR2 registers. The BNO bit controls the number of data bits which can be set to either 8 or 9, the PRT1~PRT0 bits control the choice of odd, even, mark or space parity, the PREN bit controls the parity on/off function and the STOPS bit decides whether one or two stop bits are to be used for the receiver, while the transmitter always uses two stop bits. 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 configurable for the receiver. The transmitter uses two stop bits.

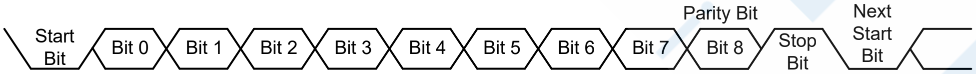
Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1 or 2
1	7	0	1	1 or 2
1	7	1	0	1 or 2
Example of 9-bit Data Formats				
1	9	0	0	1 or 2
1	8	0	1	1 or 2
1	8	1	0	1 or 2

Transmitter Receiver Data Format

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



8-bit data format



9-bit data format

UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR1 register. When the BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR_RXR register. The data to be transmitted is loaded into this TXR_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the TXR_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN bit is set, but the data will not be transmitted until the TXR_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR_RXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

Transmitting Data

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

- Make the correct selection of the BNO, PRT1~PRT0 and PREN bits to define the required word length and parity type. Two stop bits are used for the transmitter.
- Setup the BRDH, BRDL registers and UMOD2~UMOD0 bits to select the desired baud rate.
- Set the TXEN bit to ensure that the UART transmitter is enabled and the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR_RXR register. Note that this step will clear the TXIF bit.

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

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

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

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

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

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

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

Transmitting Break

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

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the BNO bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8 bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX/TX 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/TX 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/TX pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX/TX 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/TX pin, LSB first. In the read mode, the TXR_RXR register forms a buffer between the internal bus and the receiver shift register. The TXR_RXR register is a four-byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR_RXR before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERR will be subsequently indicated. For continuous multi-byte data transmission, it is strongly recommended that the receiver uses two stop bits to avoid a receiving error caused by the accumulated error of the receiver baud rate frequency.

The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNO, PRT1~PRT0, PREN and STOPS bits to define the word length, parity type and number of stop bits.
- Setup the BRDH, BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.
- Set the RXEN bit to ensure that the RX/TX pin is used as a UART receiver pin.

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

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

- The RXIF bit in the USR register will be set when the TXR_RXR register has data available, the number of the available data bytes can be checked by polling the RxCNT register content.
- When the contents of the shift register have been transferred to the TXR_RXR register and reach receiver FIFO trigger level, if the RIE bit is set, an interrupt will be generated.
- If during reception, a frame error, noise error, parity error or an overrun error has been detected, then the error flags can be set.

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

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

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

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

Idle Status

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

Receiver Interrupt

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

When a subroutine will be called with an execution time longer than the time for UART to receive five data bytes, if the UART received data could not be read in time during the subroutine execution, clear the RXEN bit to zero in advance to suspend data reception. If the UART interrupt could not be served in time to process the overrun error during the subroutine execution, ensure that both EMI and RXEN bits are disabled during this period, and then enable EMI and RXEN again after the subroutine execution has been completed to continue the UART data reception.

Managing Receiver Errors

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

Overrun Error – OERR

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

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

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

When the OERR flag is set to "1", it is necessary to read five data bytes from the four-byte deep receiver FIFO and the shift register immediately to avoid unexpected errors, such as the UART is unable to receive data. If such an error occurs, clear the RXEN bit to "0" then set it to "1" again to continue data reception.

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

Noise Error – NF

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

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

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

Framing Error – FERR

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

Parity Error – PERR

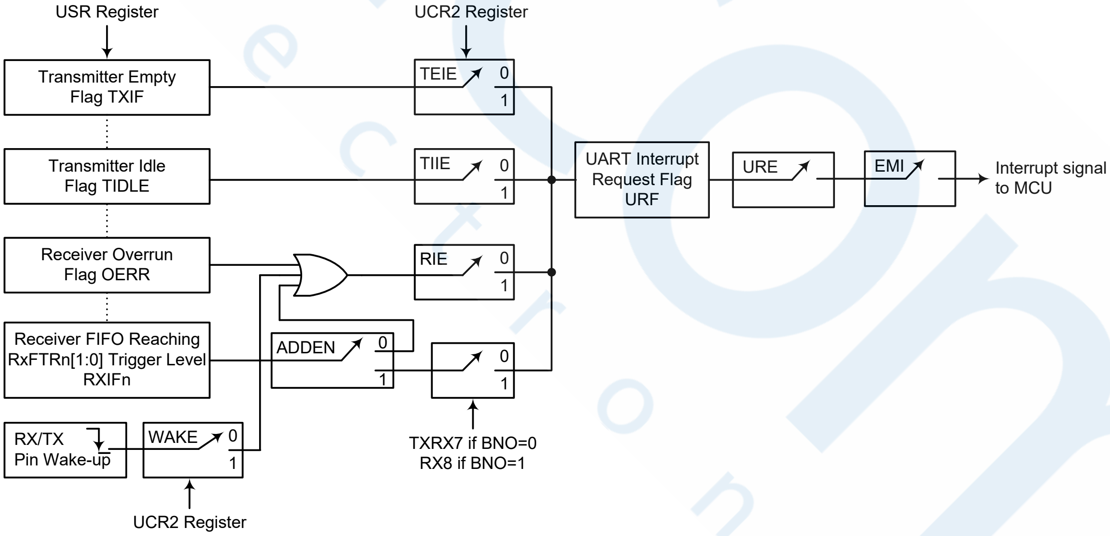
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd, even, mark or space is selected. The read only PERRn flag and the received data will be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

UART Interrupt Structure

Several individual UART conditions can generate a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and its corresponding 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 USR register flags which will generate a UART interrupt if its associated interrupt enable control bit in the UCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual UART interrupt sources.

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX/TX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the UART clock (f_H) source is switched off and the WAKE and RIE bits in the UCR2 register are set when a falling edge on the RX/TX pin occurs.

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



UART Interrupt Structure

Address Detect Mode

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

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

ADDEN Bit Function

UART Power Down and Wake-up

When the UART clock (f_H) is off, the UART will cease to function, and all clock sources to the module are shutdown. If the UART clock (f_H) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the IDLE or SLEEP Mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP Mode, note that the USR, UCR1, UCR2, UCR3, UFCR, RxCNT and TXR_RXR as well as the BRDH and BRDL regieters 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/TX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the UART clock (f_H) is off, then a falling edge on the RX/TX pin will trigger an RX/TX 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/TX pin will be ignored.

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

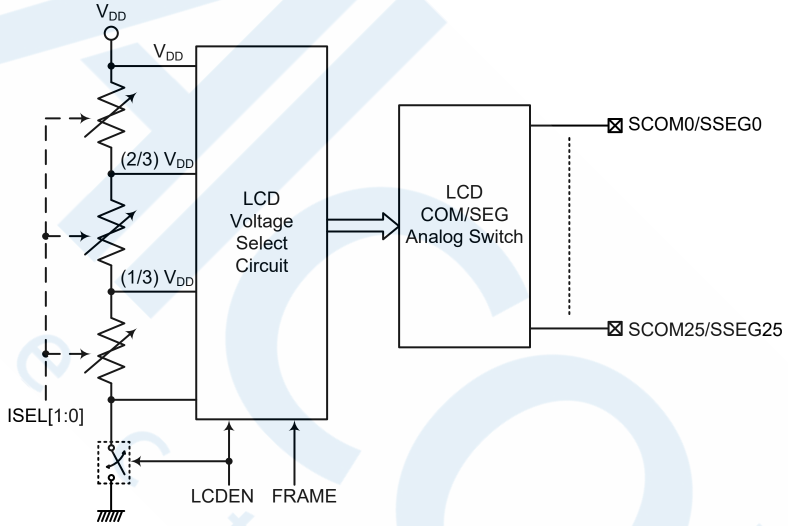
Software Controlled LCD Driver

The device has the capability of driving external LCD panels. The common pins, SCOM0~SCOMm and segment pins, SSEG0~SSEGn, where m and n are dependent upon which device is selected, for LCD driving are pin-shared with certain pins on the I/O ports. The LCD signals, COM and SEG, are generated using the application program.

LCD Operation

An external LCD panel can be driven using the device by configuring the I/O pins as common pins and segment pins. The LCD driver function is controlled using the LCD control registers which in addition to controlling the overall on/off function also controls the bias voltage setup function. This enables the LCD COM and SEG driver to generate the necessary V_{SS}, (1/3) V_{DD}, (2/3) V_{DD} and V_{DD} voltage levels for LCD 1/3 bias operation.

The LCDEN bit in the SLCDC0 register is the overall master control for the LCD driver. This bit is used in conjunction with the corresponding pin-shared function selection bits to select which I/O pins are used for LCD driving. Note that the corresponding Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation.



Software Controlled LCD Driver Structure

LCD Frames

A cyclic LCD waveform includes two frames known as Frame 0 and Frame 1 for which the following offers a functional explanation.

Frame 0

To select Frame 0, clear the FRAME bit in the SLCDC0 register to 0.

In frame 0, the COM signal output can have a value of V_{DD} or a V_{BIAS} value of $(1/3)\times V_{DD}$. The SEG signal output can have a value of V_{SS} or a V_{BIAS} value of $(2/3)\times V_{DD}$.

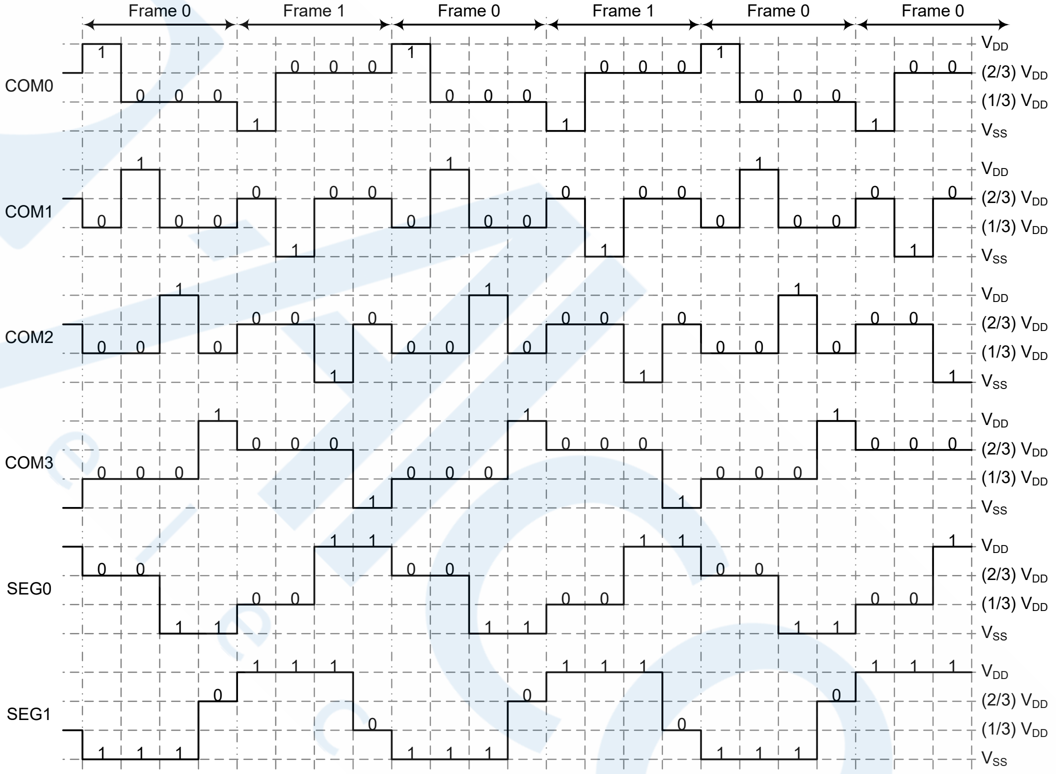
Frame 1

To select Frame 1, set the FRAME bit in the SLCDC0 register to 1.

In frame 1, the COM signal output can have a value of V_{SS} or a V_{BIAS} value of $(2/3)\times V_{DD}$. The SEG signal output can have a value of V_{DD} or a V_{BIAS} value of $(1/3)\times V_{DD}$.

The SCOMm waveform is controlled by the application program using the FRAME bit in the SLCDC0 register and the corresponding pin-shared I/O data bit for the respective SCOMm pin to determine whether the SCOMm output has a value of V_{DD}, V_{SS} or V_{BIAS}. The SSEGn waveform is controlled in a similar way using the FRAME bit and the corresponding pin-shared I/O data bit for the respective SSEGn pin to determine whether the SSEGn output has a value of V_{DD}, V_{SS} or V_{BIAS}.

The accompanying waveform diagram shows a typical 1/3 bias LCD waveform generated using the application program together with the LCD voltage select circuit. Note that the depiction of a "1" in the diagram illustrates an illuminated LCD pixel. The COM signal polarity generated on pins SCOM0 ~ SCOMm, whether "0" or "1", are generated using the corresponding pin-shared I/O data register bit.



Note: The logical values shown in the above diagram are the corresponding pin-shared I/O data bit value.

1/3 Bias LCD Waveform – 4-COM & 2-SEG Application

LCD Control Registers

The LCD SCOM and SSEG driver enables a range of selections to be provided to suit the requirement of the LCD panel which is being used. The bias resistor choice is implemented using the ISEL1 and ISEL0 bits in the SLCDC0 register. All SCOM and SSEG pins are pin-shared with I/O pins and selected as SCOMm and SSEGn pins using the corresponding pin-shared control bits and pin function selection bits.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLCDC0	FRAME	ISEL1	ISEL0	LCDEN	—	—	—	—
SLCDS0	COMSEGS7	COMSEGS6	COMSEGS5	COMSEGS4	COMSEGS3	COMSEGS2	COMSEGS1	COMSEGS0
SLCDS1	COMSEGS15	COMSEGS14	COMSEGS13	COMSEGS12	COMSEGS11	COMSEGS10	COMSEGS9	COMSEGS8
SLCDS2	COMSEGS23	COMSEGS22	COMSEGS21	COMSEGS20	COMSEGS19	COMSEGS18	COMSEGS17	COMSEGS16
SLCDS3	—	—	—	—	—	—	COMSEGS25	COMSEGS24

LCD Register Control Register List

- **SLCDC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	FRAME	ISEL1	ISEL0	LCDEN	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

Bit 7 **FRAME**: SCOMm/SSEGn Output Frame selection
0: Frame 0
1: Frame 1

Bit 6~5 **ISEL1~ISEL0**: Select resistor for R type LCD bias current (V_{DD}=5V)
00: 3×200kΩ (1/3 Bias), I_{BIAS}=8.3μA
01: 3×100kΩ (1/3 Bias), I_{BIAS}=16.6μA
10: 3×33.3kΩ (1/3 Bias), I_{BIAS}=50μA
11: 3×16.6kΩ (1/3 Bias), I_{BIAS}=100μA

Bit 4 **LCDEN**: LCD control bit
0: Off
1: On

When the LCDEN bit is cleared to 0, then the SCOMm and SSEGn outputs will be fixed at a V_{SS} level.

Bit 3~0 Unimplemented, read as "0"

- **SLCDS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	COMSEGS7	COMSEGS6	COMSEGS5	COMSEGS4	COMSEGS3	COMSEGS2	COMSEGS1	COMSEGS0
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 **COMSEGS7**: SCOM7/SSEG7 pin function selection
0: SCOM7
1: SSEG7

Bit 6 **COMSEGS6**: SCOM6/SSEG6 pin function selection
0: SCOM6
1: SSEG6

Bit 5 **COMSEGS5**: SCOM5/SSEG5 pin function selection
0: SCOM5
1: SSEG5

Bit 4 **COMSEGS4**: SCOM4/SSEG4 pin function selection
0: SCOM4
1: SSEG4

Bit 3 **COMSEGS3**: SCOM3/SSEG3 pin function selection
0: SCOM3
1: SSEG3

Bit 2 **COMSEGS2**: SCOM2/SSEG2 pin function selection
0: SCOM2
1: SSEG2

Bit 1 **COMSEGS1**: SCOM1/SSEG1 pin function selection
0: SCOM1
1: SSEG1

Bit 0 **COMSEGS0**: SCOM0/SSEG0 pin function selection
0: SCOM0
1: SSEG0

- **SLCDS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	COMSEGS15	COMSEGS14	COMSEGS13	COMSEGS12	COMSEGS11	COMSEGS10	COMSEGS9	COMSEGS8
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 **COMSEGS15**: SCOM15/SSEG15 pin function selection
0: SCOM15
1: SSEG15

Bit 6 **COMSEGS14**: SCOM14/SSEG14 pin function selection
0: SCOM14
1: SSEG14

Bit 5 **COMSEGS13**: SCOM13/SSEG13 pin function selection
0: SCOM13
1: SSEG13

Bit 4 **COMSEGS12**: SCOM12/SSEG12 pin function selection
0: SCOM12
1: SSEG12

Bit 3 **COMSEGS11**: SCOM11/SSEG11 pin function selection
0: SCOM11
1: SSEG11

Bit 2 **COMSEGS10**: SCOM10/SSEG10 pin function selection
0: SCOM10
1: SSEG10

Bit 1 **COMSEGS9**: SCOM9/SSEG9 pin function selection
0: SCOM9
1: SSEG9

Bit 0 **COMSEGS8**: SCOM8/SSEG8 pin function selection
0: SCOM8
1: SSEG8

- **SLCDS2 Register**

Bit	7	6	5	4	3	2	1	0
Name	COMSEGS23	COMSEGS22	COMSEGS21	COMSEGS20	COMSEGS19	COMSEGS18	COMSEGS17	COMSEGS16
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 **COMSEGS23**: SCOM23/SSEG23 pin function selection
0: SCOM23
1: SSEG23

Bit 6 **COMSEGS22**: SCOM22/SSEG22 pin function selection
0: SCOM22
1: SSEG22

Bit 5 **COMSEGS21**: SCOM21/SSEG21 pin function selection
0: SCOM21
1: SSEG21

Bit 4 **COMSEGS20**: SCOM20/SSEG20 pin function selection
0: SCOM20
1: SSEG20

Bit 3 **COMSEGS19**: SCOM19/SSEG19 pin function selection
0: SCOM19
1: SSEG19

Bit 2 **COMSEGS18**: SCOM18/SSEG18 pin function selection
0: SCOM18
1: SSEG18

Bit 1 **COMSEGS17**: SCOM17/SSEG17 pin function selection
0: SCOM17
1: SSEG17

Bit 0 **COMSEGS16**: SCOM16/SSEG16 pin function selection
0: SCOM16
1: SSEG16

- **SLCDS3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	COMSEGS25	COMSEGS24
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1 **COMSEGS25**: SCOM25/SSEG25 pin function selection
0: SCOM25
1: SSEG25

Bit 0 **COMSEGS24**: SCOM24/SSEG24 pin function selection
0: SCOM24
1: SSEG24

Low Voltage Detector – LVD

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

LVD Register

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

- **LVDC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	—	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6 Unimplemented, read as "0"

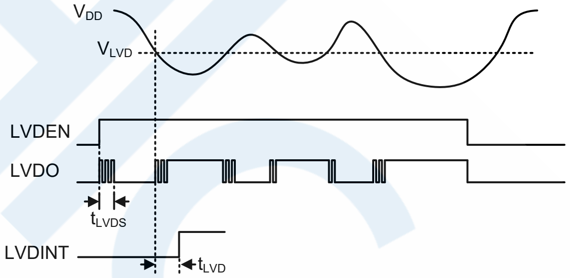
Bit 5 **LVDO**: LVD Output Flag
0: No Low Voltage Detected
1: Low Voltage Detected

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

Bit 3	Unimplemented, read as "0"
Bit 2~0	**VLVD2~VLVD0**: LVD voltage selection 000: 1.8V 001: 2.0V 010: 2.4V 011: 2.7V 100: 3.0V 101: 3.3V 110: 3.6V 111: 4.0V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 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.



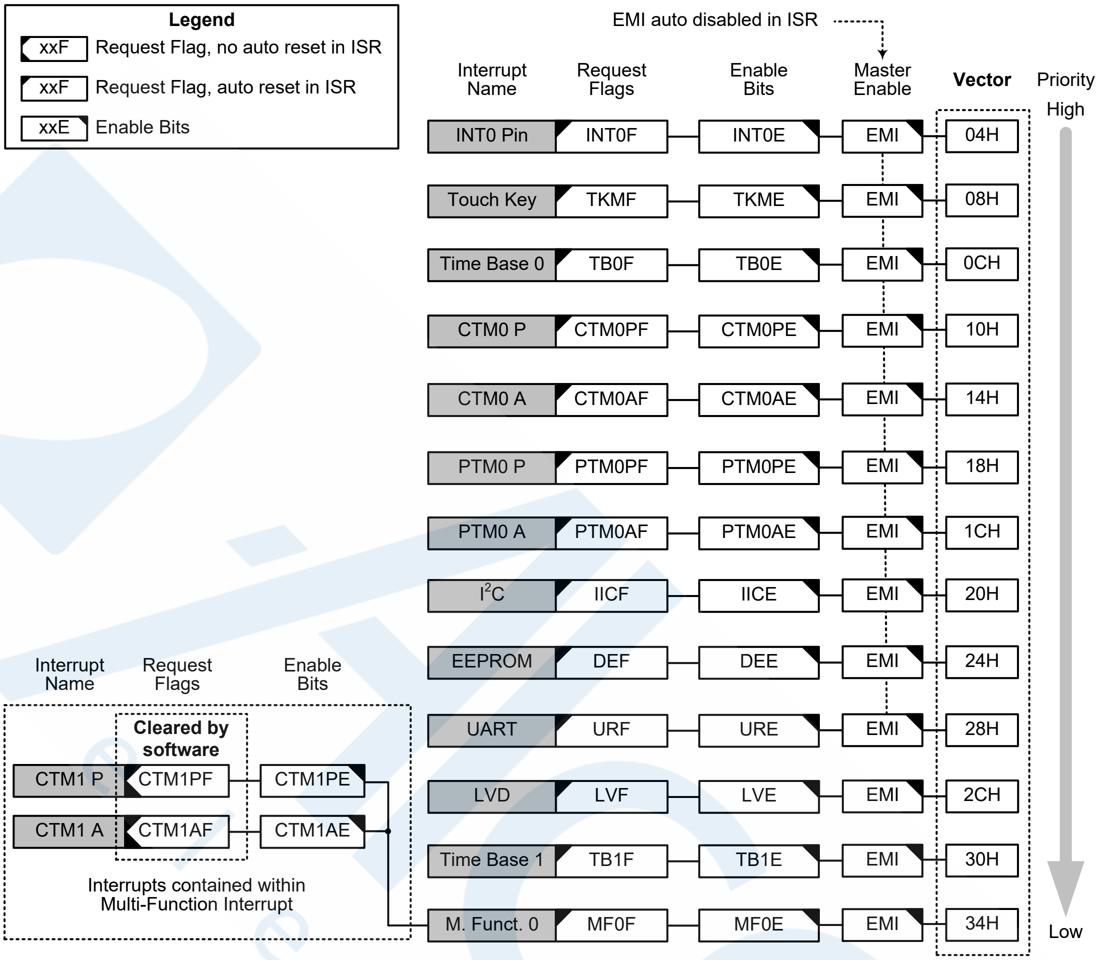
LVD Operation

The Low Voltage Detector also has its own interrupt, 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, i.e., V_{DD} falls below the preset LVD voltage. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated. This will cause the device to wake-up from the IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before the device enters the IDLE Mode.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module requires microcontroller attention, the 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 interrupt is generated by the action of the external INT0 pin, while the internal interrupts are generated by various internal functions such as the TMs, LVD and Time bases, etc.

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.



Interrupt Structure

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers falls into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0 register which setups the Multi-function interrupt. Finally there is an INTEG register to set the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INT0 Pin	INT0E	INT0F	—
Touch Key Module	TKME	TKMF	—
Time Bases	TBnE	TBnF	n=0~1

Function	Enable Bit	Request Flag	Notes
CTM	CTMnPE	CTMnPF	n=0~1
	CTMnAE	CTMnAF	
PTM	PTM0PE	PTM0PF	—
	PTM0AE	PTM0AF	
I²C Interface	IICE	IICF	—
EEPROM	DEE	DEF	—
UART	URE	URF	—
LVD	LVE	LVF	—
Multi-function	MF0E	MF0F	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INT0S1	INT0S0
INTC0	—	TB0F	TKMF	INT0F	TB0E	TKME	INT0E	EMI
INTC1	PTM0AF	PTM0PF	CTM0AF	CTM0PF	PTM0AE	PTM0PE	CTM0AE	CTM0PE
INTC2	LVF	URF	DEF	IICF	LVE	URE	DEE	IICE
INTC3	—	—	MF0F	TB1F	—	—	MF0E	TB1E
MFI0	—	—	CTM1AF	CTM1PF	—	—	CTM1AE	CTM1PE

Interrupt Register List

• INTEG Register

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **INT0S1~INT0S0**: Interrupt trigger edge selection 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	—	TB0F	TKMF	INT0F	TB0E	TKME	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 **TB0F**: Time Base 0 request flag
0: No request
1: Interrupt request

Bit 5 **TKMF**: Touch Key Module interrupt request flag
0: No request
1: Interrupt request

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

Bit 3 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

Bit 2 **TKME**: Touch Key Module interrupt control
0: Disable
1: Enable

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

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

- **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTM0AF	PTM0PF	CTM0AF	CTM0PF	PTM0AE	PTM0PE	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 **PTM0AF**: PTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **PTM0PF**: PTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 5 **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 **PTM0AE**: PTM0 Comparator A match interrupt control
0: Disable
1: Enable

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

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

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

• INTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	LVF	URF	DEF	IICF	LVE	URE	DEE	IICE
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 **LVF**: LVD interrupt request flag
0: No request
1: Interrupt request

Bit 6 **URF**: UART interrupt request flag
0: No request
1: Interrupt request

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

Bit 4 **IICF**: I^{2}C interrupt request flag
0: No request
1: Interrupt request

Bit 3 **LVE**: LVD interrupt control
0: Disable
1: Enable

Bit 2 **URE**: UART interrupt control
0: Disable
1: Enable

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

Bit 0 **IICE**: I^{2}C interrupt control
0: Disable
1: Enable

• INTC3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	MF0F	TB1F	—	—	MF0E	TB1E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as “0”

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

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

Bit 3~2 Unimplemented, read as “0”

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

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

- **MFI0 Register**

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

Bit 7~6 Unimplemented, read as “0”

Bit 5 **CTM1AF**: CTM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4 **CTM1PF**: CTM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2 Unimplemented, read as “0”

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

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

Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A match 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.

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

External Interrupt

The external interrupt is controlled by signal transitions on the INT0 pin. An external interrupt request will take place when the external interrupt request flag, INT0F, is set, which will occur when a transition appears on the external interrupt pin. 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. 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, must first be set. As the external interrupt pin is pin-shared with I/O pin, it should be configured as external interrupt pin before the external pin interrupt function is enabled. The pin must also be setup as an input type 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 flag, INT0F, 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 pin 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 is a Multi-function interrupt. Unlike the other independent interrupts, this interrupt has no independent source, but rather are formed from other existing interrupt sources, namely the CTM1 Interrupts.

A Multi-function interrupt request will take place when the Multi-function interrupt request flag, MF0F, is set. The Multi-function interrupt flag will be set when any of its included functions generate an interrupt request flag. When the Multi-function interrupt is enabled and the stack is not full, and either one of the CTM1 interrupts contained within the Multi-function interrupt occurs, a subroutine call to the Multi-function interrupt vector will take place. When the interrupt is serviced, the 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 flag will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupt will not be automatically reset and must be manually reset by the application program.

TM Interrupts

The Compact and Periodic Type TMs each have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. The CTM0 and PTM0 interrupt sources have their own individual vector while the CTM1 interrupt is contained within the Multi-function Interrupt. For 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 the CTM0 or PTM0 interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and the CTM0 or PTM0 comparator match situation occurs, a subroutine call to the CTM0 or PTM0 Interrupt vector location, will take place. When the TM interrupt is serviced, the CTM0 or PTM0 interrupt request flag will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

To allow the program to branch to the CTM1 interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and the relevant Multi-function Interrupt enable bit, MF0E, must first be set. When the interrupt is enabled, the stack is not full and the CTM1 comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector location, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, the related MF0F flag will also be automatically cleared. As the CTM1 interrupt request flag will not be automatically cleared, they have to be cleared by the application program.

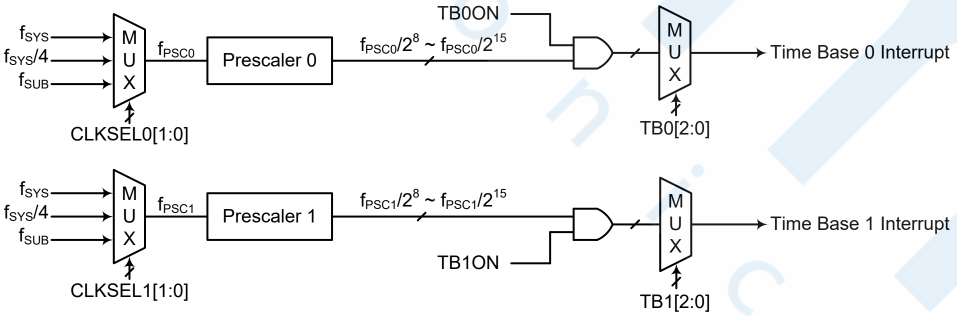
Touch Key Module Interrupt

A Touch Key interrupt request will take place when the Touch Key interrupt request flag, TMKF, is set, which occurs when the time slot counter in all the touch key modules or in touch key module 0 overflows. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Touch Key interrupt enable bit, TKME, must be first set. When the interrupt is enabled, the stack is not full and the Touch Key time slot counter overflow occurs, a subroutine call to the interrupt vector, will take place. When the interrupt is serviced, the Touch Key interrupt request flag will be automatically reset and 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 signals in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupts is to provide an interrupt signal at fixed time periods. Their respective clock source, f_{PSC0} or f_{PSC1}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C or TB1C register to obtain longer interrupt periods whose value ranges. The clock source that generates f_{PSC0} or f_{PSC1}, which in turn controls the Time Base interrupt period, is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the PSC0R and PSC1R register respectively.



Time Base Interrupts

- **PSC0R Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL01	CLKSEL00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSEL01~CLKSEL00**: Prescaler 0 clock source selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

- **PSC1R Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL11	CLKSEL10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSEL11~CLKSEL10**: Prescaler 1 clock source selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

- **TB0C Register**

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

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

Bit 6~3 Unimplemented, read as “0”

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

- **TB1C Register**

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

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

Bit 6~3 Unimplemented, read as “0”

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

I²C Interrupt

An I²C interrupt request will take place when the I²C Interrupt request flag, IICF, is set, which occurs when a byte of data has been received or transmitted by the I²C interface, or an I²C slave address match occurs, or an I²C bus time-out occurs. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the I²C Interrupt enable bit, IICE, 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 I²C Interrupt vector, will take place. When the interrupt is serviced, the I²C Interrupt flag, IICF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

UART Interrupt

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

LVD 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 interrupt vector address, the global interrupt enable bit, EMI, and Low Voltage Interrupt enable bit, LVE, 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 LVD Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the LVD Interrupt flag, LVF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

EEPROM Interrupt

An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Erase or Write cycle ends. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Erase or Write cycle ends, a subroutine call to the EEPROM Interrupt vector will take place. When the EEPROM Interrupt is serviced, the EEPROM Interrupt flag, DEF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

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 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 the Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flag, MF0F, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

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

Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

NFC Reader

The device includes a fully integrated NFC reader. This chapter will introduce the NFC reader in terms of Communication Interface Function, Functional Description, Register Description and Command Set Description.

Communication Interfaces

Overview

The NFC reader supports the I^2C and serial UART interfaces. After reset, the NFC reader identifies the current host interface type automatically by checking the logic level on the pins. The following table lists the different connection configurations.

Pin	Interfae Type	
	UART	I^2C
SDA_NFC/RX_NFC	RX_NFC	SDA_NFC
I2C	0	1
SCL_NFC/TX_NFC	TX_NFC	SCL_NFC

UART Interface

The UART default transfer speed is 9.6kbit/s. To modify the transfer speed, the host controller must write a new value to the SerialSpeed register, where the BR_T0[2:0] and BR_T1[4:0] bits define the factors for setting the transfer speed. The host controller needs to first configure the register using a data rate of 9.6kbit/s, and then adjusts its data rate corresponding to the new register value.

The BR_T0[2:0] and BR_T1[4:0] settings are described below.

BR_Tn	Bit 0	Bit 1	Bit 2	Bit 3	Bit4	Bit 5	Bit6	Bit 7
BR_T0 factor	1	1	2	4	8	16	32	64
BR_T1 range	1 to 32	33 to 64	33 to 64	33 to 64	33 to 64	33 to 64	33 to 64	33 to 64

BR_T0 and BR_T1 Settings

Examples of different transfer speeds and the relevant register settings are given in the following table.

Transfer Speed (kbit/s)	7.2	9.6	14.4	19.2	38.4	57.6	115.2	128	230.4	460.8	921.6	1228.8
SerialSpeed register value	FAh	EBh	DAh	CBh	ABh	9Ah	7Ah	74h	5Ah	3Ah	1Ch	15h
Transfer speed accuracy	-0.25	0.32	-0.25	0.32	0.32	-0.25	-0.25	-0.06	-0.25	-0.25	1.45	0.32

Selectable UART Transfer Speeds

Note: Transfer speeds exceeding 1228.8kbit/s are not supported.

The transfer speeds shown in the above table are calculated according to the following equations.

- If BR_T0[2:0]=0

$$\text{Transfer Speed} = (27.12\times10^6) \div (\text{BR\_T0}+1)$$

- If BR_T0[2:0]>0

$$\text{Transfer Speed} = (27.12\times10^6) \div [(\text{BR\_T1}+33) \div 2^{(\text{BR\_T0}-1)}]$$

The UART frame format is given in this table. The LSB of the data and address bytes must be sent first. No parity bit is used during transmission.

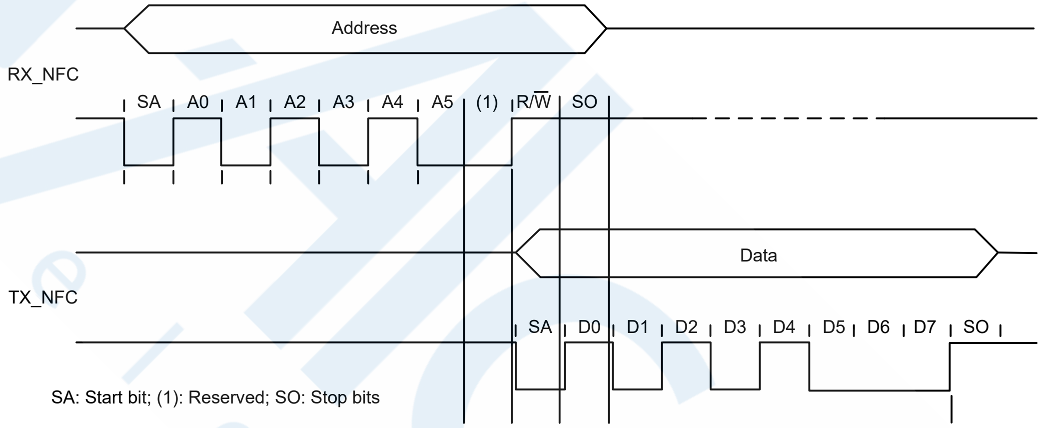
Bit	Length	Value
Start	1-bit	0
Data	8-bit	data
Stop	1-bit	1

UART Frame Format

To read out data using the UART interface, the procedure shown in the following table must be used. The first sent byte defines both mode and address.

Pin	Byte 0	Byte 1
RX_NFC	Address	—
TX_NFC	—	Data 0

Byte Order for Reading Data

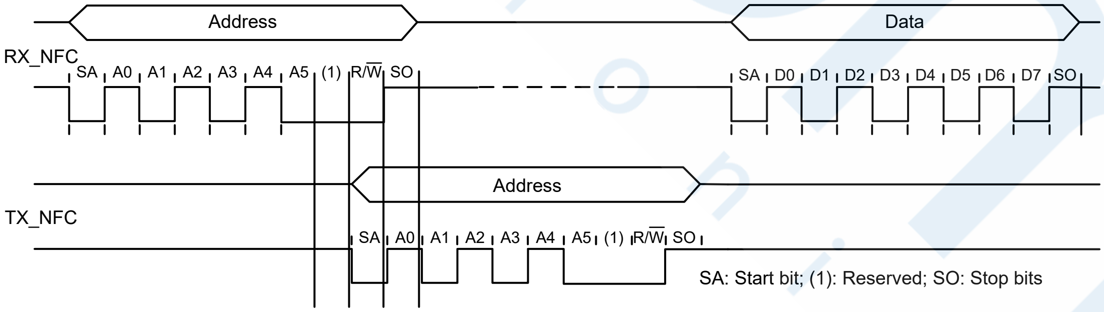


Timing Diagram for Reading Data

To write data using the UART interface, the procedure shown in the following table must be used. The first sent byte defines both mode and address.

Pin	Byte 0	Byte 1
RX_NFC	Address 0	Data 0
TX_NFC	—	Address 0

Byte Order for Writing Data



Timing Diagram for Writing Data

The address byte must meet the following format, as shown in the table below.

- The MSB of the first byte defines the mode.
  - MSB=0 for write mode
  - MSB=1 for read mode
- Bit6 is reserved.
- Bit[5:0] defines the address.

Bit7 (MSB)	Bit 6	Bit 5	Bit4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
1 or 0	Reserved	Address	Address	Address	Address	Address	Address

Address Byte

I^{2}C Interface

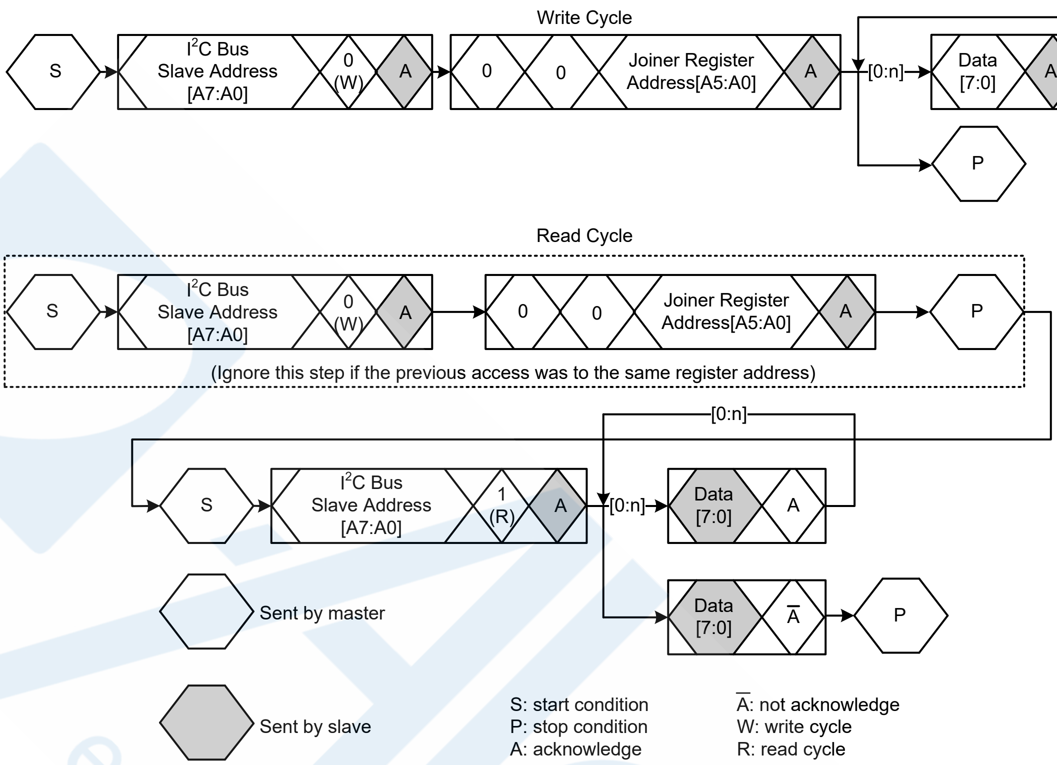
The NFC reader acts as a slave during I^{2}C communication and the implemented interface is inaccordance with the I^{2}C standard. The data transfer speed is up to 400kbit/s.

To write data from the host controller using the I^{2}C bus, the following procedure and format must be followed:

- The first byte indicates the device address in accordance with the I^{2}C bus standard.
- The second byte indicates the register address, followed by up to n data bytes.
- The Read/Write bit is 0.

To read data from the NFC reader using the I^{2}C bus, the following procedure and format must be followed:

- The first byte indicates the device address in accordance with the I^{2}C bus standard.
- The second byte indicates the register address. No data bytes are added.
- The Read/Write bit is 0.
- After the write access, read access can be started. The host sends the device address. In response, the contraceless chip sends the content of the accessed register to the host.
- The Read/Write bit is 1.



Register Read and Write Access

The address byte must meet the following format:

- The I²C device address bits are fixed at 0101000.
- The LSB defines the mode.
  - LSB=0, write data to the NFC reader.
  - LSB=1, read data from the NFC reader.

Functional Description

CRC Coprocessor

The following CRC coprocessor parameters can be configured:

- The CRC preset value can be either 0000h, 6363h, A671h or FFFFh depending on the CRCPreset[1:0] setting in the Mode register.
- The CRC polynomial for the 16-bit CRC is $X^{16} + X^{12} + X^5 + 1$.
- The CRCResult_H and CRCResult_L registesrs indicate the CRC calculation result high byte and low byte respectively.
- The MSBFirst bit in the Mode register indicates that data will be loaded with the MSB first.

Parameter	Value
CRC register length	16-bit CRC
CRC algorithm	Algorithm according to ISO/IEC 14443 A and ITU-T
CRC preset value	0000h, 6363h, A671h or FFFFh depending on CRCPreset[1:0] setting in the Mode register

FIFO Buffer

The NFC reader contains an 8×64 FIFO buffer. It is used to cache the input and output data streams for communication between the host and the NFC reader internal state machine. It can manage up to 64 bytes of data streams without considering timing limitations.

Accessing the FIFO Buffer

The FIFO buffer input and output data bus is connected to the FIFOData register. Writing to the FIFOData register will store one byte in the FIFO buffer and increase the internal FIFO buffer write pointer by 1. Reading from the FIFOData register will read the data stored in the pointed FIFO buffer and decrease the FIFO buffer read pointer by 1. The distance between the FIFO buffer write and read pointers can be obtained by reading the FIFOLevel register.

When the host controller sends a command, the NFC reader can access the FIFO buffer according to the command while the command is in progress. The FIFO buffer can continue to be used for input and output only when it is valid. The host controller must ensure that there are no unintentional accesses to the FIFO buffer.

Controlling the FIFO Buffer

The FIFO buffer pointers can be reset by setting the FlushBuffer bit in the FIFOLevel register to logic 1. Consequently, the FIFOLevel[6:0] bits are all set to logic 0 and the BufferOvfl flag in the Error register is cleared to zero. In this situation, the stored bytes can no longer be accessible, allowing the FIFO buffer to store another 64 data bytes.

FIFO Buffer State Information

The host device can obtain the following FIFO buffer state information:

- The number of data bytes stored in the FIFO buffer: FIFOLevel[6:0] bits in the FIFOLevel register
- FIFO buffer almost full warning: HiAlert bit in the Status1 register
- FIFO buffer almost empty warning: LoAlert bit in the Status1 register
- FIFO buffer overflow warning: BufferOvfl bit in the Error register, which can be cleared only by setting the FlushBuffer bit in the FIFOLevel register

The NFC reader can generate an interrupt signal when any one of the following conditions occurs:

- The LoAlertIEn bit in the ComIEn register is set to logic 1, it actives the IRQ_NFC pin when the LoAlert bit in the Status1 register changes to logic 1.
- The HiAlertIEn bit in the ComIEn register is set to logic 1, it actives the IRQ_NFC pin when the HiAlert bit in the Status1 register changes to logic 1.
- If the WaterLevel[5:0] value set in the WaterLevel register is greater than or equal to the remaining space in the FIFO buffer, the HiAlert bit is set to logic 1:

  HiAlert=1 if (64-FIFOLength)≤WaterLevel
- If the WaterLevel[5:0] value set in the WaterLevel register is greater than or equal to the used space in the FIFO buffer, the LoAlert bit is set to logic 1:

  LoAlert=1 if FIFOLength≤WaterLevel

Interrupt Request System

The NFC reader indicates various interrupt events by setting the IRq bit in the Status1 register or by activing the IRQ_NFC pin. The signal on the IRQ_NFC pin can be used to interrupt the host to use its interrupt handling capabilities. This greatly improves the efficiency of host software execution.

Interrupt Source Overview

The following table lists the available interrupt flags, the corresoponding interrupt source and the trigger condition for their activation.

The TimerIRq interrupt flag in the ComIrq register being set high indicates an interrupt generated by the timer. When the timer decreases from 1 to 0, this flag will be set high.

The TxIRq flag in the ComIrq register being set high indicates that the transmitter sending has finished. When the state changes from sending data to sending the end of the frame, the transmitter will automatically set the corresponding interrupt flag high. The CRC coprocessor will set the CRCIRq flag high in the DivIrq register after processing all the data in the FIFO buffer which is indicated by the CRCReady bit.

The RxIRq flag in the ComIrq register being set high indicates that the end of data reception has been detected.

The IdleIRq flag in the ComIrq register will be set high if a command execution has finished and the Command[3:0] field in the Command register has changed to the Idle state.

When the HiAlert bit is set to 1 and the HiAlertIRq flag in the ComIrq register is set to 1, it indicates that the remaining space in the FIFO buffer has reached the level indicated by the WaterLevel[5:0] bits. When the LoAlert bit is set to 1 and the LoAlertIRq flag in the ComIrq register is set to 1, it indicates that the used space in the FIFO buffer has reached the level indicated by the WaterLevel[5:0] bits.

The ErrIRq flag in the ComIrq register being set high indicates that an error has been detected during the contractless UART transmission and reception. This is indicated when any bit in the Error register is set to logic 1.

The TagDetIRq flag in the DivIrq register being set high indicates that in the LPCD mode an external contractless card has been detected.

Intrrupt Flag	Interrupt Source	Trigger Action
TimerIRq	Timer	The timer counts from 1 to 0
TxIRq	Transmitter	A data stream transmitting ends
CRCIRq	CRC coprocessor	All data in the FIFO buffer has been processed
RxIRq	Receiver	A data stream receiving ends
IdleIRq	ComIrq register	Command execution ends
HiAlertIRq	FIFO buffer	FIFO buffer is almost full
LoAlertIRq	FIFO buffer	FIFO buffer is almost empty
ErrIRq	Contractless UART	An error is detected
TagDetIRq	LPCD trigger	In the LPCD mode, a card enters the atenna effective range

Timer Unit

The NFC reader has a timer unit which can be used by the external host to manage timing tasks. The timer unit can be used in one of the following timer/counter configurations:

- Time-out counter
- Watchdog counter
- Stopwatch
- Programmable one-shot trigger
- Periodic trigger

The timer unit can be used to measure the time interval between two events or to indicate the occurrence of a specified event after a certain period of time. The timer can be triggered by events explained in the following paragraphs. The timer unit does not affect any internal events, for example, a timer time-out event during data reception does not affect the automatic processing of this process. In addition, several timer-related bits can be used to generate an interrupt.

The timer has an input clock of 13.56MHz, which is obtained by dividing the frequency of a 27.12MHz quartz crystal oscillator. The timer consists of two stages: prescaler and counter.

The prescaler (TPrescaler) is a 12-bit counter, which is set using the TPrescaler_Hi[3:0] bits in the TMode register and the TPrescaler_Lo[7:0] bits in the TPrescaler register. The 16-bit reload value for the counter, TReloadVal, can be set between 0 and 65535 using the TReload_H and TReload_L registers. The current value of the timer is indicated in the TCouterVal_H and TCouterVal_L registers.

When the counter value reaches 0, an interrupt is automatically generated, which is indicated by the TimerIRq flag in the ComIrq register. If enabled, this interrupt signal can be indicated on the IRQ_NFC pin. The TimerIRq flag can be set and reset by the host. Depending on the configuration, the timer can stop at 0 or restart with the value set in the TReload_H and TReload_L registers.

The timer status is indicated by the TRunning bit in the Status1 register.

The timer can be started using the TStartNow bit in the Control register and stopped using the TStopNow bit in the same register.

To meet some specific protocol requirements, the timer can also be activated automatically by setting the TAuto bit in the TMode register to logic 1.

The delay time of a timing process is set by the reload value plus one. The total delay time is calculated using the following equation:

t_d=(TPrescaler×2+1)×(TReloadVal+1)÷13.56MHz

An example of calculating the total delay time is shown below, where TPrescaler=4095 and TReloadVal=65535:

39.59s=(4095×2+1)×(65535+1)÷13.56MHz

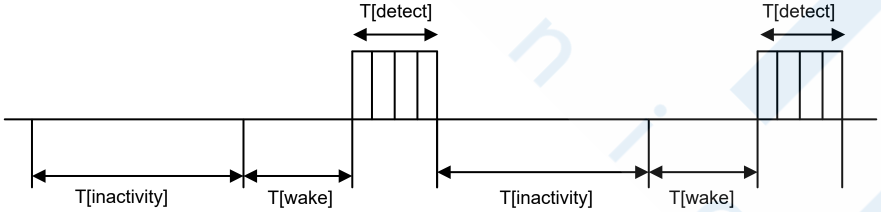
For example, a delay unit of 25μs requires 339 timer clock cycles to be counted and a TPrescaler value of 169. This configuration allows the timer to count every 25μs up to 65535 times.

LPCD Unit

The LPCD is the abbreviation of Low Power external Card Detection. This function allows the NFC reader to detect whether an external contactless card is close to it with a very low standby power consumption. If an card has been detected, an interrupt can be output to inform the host MCU to enter the normal reader communication mode and implement data exchange with the card.

LPCD Operating Principle

The LPCD mode is divided into three stages, T[inactivity](sleep period), T[wake](active period) and T[detect](detection period), acorresponding to different actions, as shown below.



In the initial state, the chip detects the antenna field strength in the current field and automatically records it. When a card is within the effective range of the antenna, it will cause the antenna field strength to change beyond the Delta[3:0] value in the LPCD register, then the chip will automatically change to the reader state and trigger the TagDetIRq interrupt. The LPCD function related registers include the LPCD, WUPeriod and SwingsCnt.

The LPCD time parameters are described below:

- Sleep period: T[inactivity]=WUPeriod[7:0] × 256 × Tclk_32k
- Active period: T[wake]=400μs (Typ.)
- Detection period: T[detect]=SwingsCnt[3:0] × 16 × 4 × Tclk_27M12
- Card detection total time=(T[inactivity] + T[wake] + T[detect] × (Skip[2:0] + 1)

TX Driver

The signals on the TX1A and TX2A pins are signals modulated by a 13.56MHz carrier, which can be used to directly drive an antenna with some passive components for matching and filtering.

The signals on the TX1A and TX2A pins can be configured using the TxControl register. The modulation index can be set by adjusting the impedance of the driver so as to adjust the output power, current consumption and operating distance. The impedance of the P driver is configured using the CWGsP and ModGsP registesrs. The impedance of the N driver is configured using the GsN register. The modulation index also depends on antenna design and tunning.

Power Saving Modes

Hardware Power-down

The hardware power-down mode is enabled when the NRSTPD pin is pulled low. This mode turns off all internal current sinks including the oscillator. All digital input buffers are separated from the input pins and their functionality is turned off (except pin NRSTPD). The output pins remain at either a high or low level.

Software Power-down

The software power-down mode is entered immediately after the PowerDown bit in the Command register is set to logic 1. This mode turns off all internal current sinks including the oscillator buffer. However, the digital input buffers are not separated from the input pins and remain their functionality. The status of the digital output pins remains unchanged.

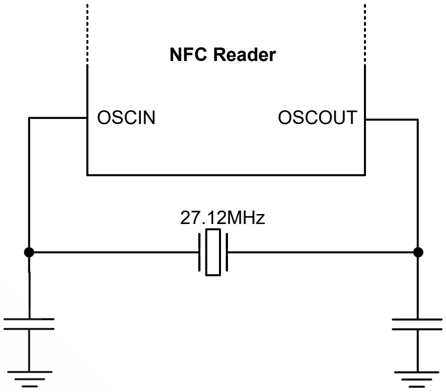
During the software power-down, all register values, FIFO buffer content and configuration remain unchanged.

After setting the PowerDown bit to logic 0, it takes 1024 clocks to exit the software power-down mode. Setting the PowerDown bit low does not immediately clear it. This bit is cleared automatically by the chip when the software power-down mode is exited.

Transmitter Power-down

The transmitter power-down mode turns off the internal antenna drivers, thereby switching off the RF field. The transmitter power-down mode is entered by setting the TxRFEn bit in the TxControl register to 00b.

Oscillator Circuit



Quatz Crystal Connection

The clock applied to the NFC reader provides a time basis for the encoder and decoder synchronized to the system. Therefore, the stability of the clock frequency is an important factor to ensure the good performance of the system. To achieve the best performance, clock jitter must be as minimized.

Registers

Register Bit Behavior

Depending on the functionality of the register, the access conditions of the registers are also different. The following table describes different access conditions for the registers.

Abbreviation	Behavior	Description
R/W	Read/Write	These bits can be read and written by the host controller. Since they are used only for control, their contents are not affected by the internal state machine. For example, the ComIEn register can be read and written by the host controller, but it can only be read by the internal state machine and cannot be changed by it.
D	Dynamic	These bits can be read and written by the host controller. They can also be written automatically by the internal state machine. For example, the Command register automatically changes the value of certain bits in it after command execution.
W	Write only	Reading these register bits always returns zero.
R	Read only	These registser bits are determined by the internal states only.
Reserved	—	These register bits are reserved for future use and cannot be changed.

Register Overview

To access the registers of pages 4~6, the PageSel register of page 3 should first be configured to the correct switch value before executing any operations to the registers in the corresponding page.

Address (Hex)	Register Name	Function
Page 0		
00h	Reserved	—
01h	Command	Start and stop command execution
02h	ComIEn	Interrupt request enable or disable control bits
03h	DivIEn	Interrupt request enable or disable control bits
04h	ComIrq	Interrupt request bits
05h	DivIrq	Interrupt request bits
06h	Error	Error status of the last command executed
07h	Status1	Communication status bits

Address (Hex)	Register Name	Function
08h	Status2	Receiver and transmitter status bits
09h	FIFOData	64-byte FIFO buffer input/output
0Ah	FIFOLevel	Number of bytes stored in the FIFO buffer
0Bh	WaterLevel	Level for FIFO buffer overflow and empty warning
0Ch	Control	Internal controller
0Dh	BitFraming	Adjustment for bit-oriented frames
0Eh	Coll	Bit position of the first bit-collision detected
Page 1		
10h	Reserved	—
11h	Mode	General mode setting for transmitting and receiving
12h	TxMode	Transmission data rate and frame format
13h	RxMode	Reception data rate and frame format
14h	TxControl	Antenna driver pins TX1A and TX2A control
15h	TxASK	Transmission modulation settings
16h	TxSel	Antenna driver signal source selection
17h	RxSel	Internal receiver settings
18h	Reserved	—
19h	Demod	Demodulation settings
1Ch	MfTx	Transmission waiting time control
1Dh	MfRx	Parity function and high-pass bandwidth settings
1Eh	TypeB	ISO/IEC 14443 B control
1Fh	SerialSpeed	Serial UART data rate selection
Page 2		
20h	Reserved	—
21h	CRCResult_H	CRC result high byte
22h	CRCResult_L	CRC result low byte
23h	Reserved	—
24h	ModWidth	Modulation width control
25h	Reserved	—
26h	RFCfg	Receiver gain setting
27h	GsN	Controls the conductance of N driver output during non-modulation period and modulation period
28h	CWGsP	Controls the conductance of P driver output during non-modulation period
29h	ModGsP	Controls the conductance of P driver output during modulation period
2Ah	TMode	Internal timer settings
2Bh	TPrescaler	
2Ch	TReload_H	16-bit timer reload value high byte
2Dh	TReload_L	16-bit timer reload value low byte
2Eh	TCounterVal_H	16-bit timer current value high byte
2Fh	TCounterVal_L	16-bit timer current value low byte
Page 3		
37h	PageSel	Controls the register page switching
Page 4		
33h	RxAlgorithm0	Demodulation algorithm adjustment
34h	AGCCfg0	Automatic gain control bits
35h	AGCCfg1	Automatic gain control bits
36h	RxAlgorithm1	Demodulation algorithm adjustment

Address (Hex)	Register Name	Function
38h	RxAlgorithm2	Demodulation algorithm adjustment
39h	RxAlgorithm3	Demodulation algorithm adjustment
3Ah	RxCK	TypeA waveform falling time adjustment; phase selection
3Bh	RxBand	Signal-to-noise ratio adjustment
3Ch	LPCD	LPCD control bits
3Dh	WUPeriod	LPCD sleep time setting
3Eh	SwingsCnt	LPCD enable; LPCD detection time setting
3Fh	Special	Receiver demodulation control
Page 5		
31h	Analog	High temperature protection control bits
32h	Noise	Noise threshold control bits
33h	StepCtrl	Transmitter modulation control bits
34h	AgcMin	AGC amplitude threshold setting
38h	RxAlgorithm6	Demodulation algorithm adjustment
39h	RxAlgorithm7	Demodulation algorithm adjustment
3Ah	RxAlgorithm8	Demodulation algorithm adjustment
3Bh	RxAlgorithm9	Demodulation algorithm adjustment
Page 6		
31h	LPCDRef	LPCD reference value
32h	LPCDDet	LPCD detected value
33h	Calibration	LPCD calibration control bits
34h	RC27MCalValue	LPCD 27.12MHz clock calibration value
35h	RC32KCalValue	LPCD 32kHz clock calibration value
36h	LPCDADCRef	LPCD ADC reference level
38h	CWGsN_LPCD	LPCD N driver control
39h	CWGsP_LPCD	LPCD P driver control

Register Description

Public Registesr description

Page 0

• Command Register

Start and stop command execution.

Address	Bit	7	6	5	4	3	2	1	0
01h	Name	Reserved		RcvOff	PowerDown	Command[3:0]			
	Type	R		R/W	R/W	R/W			
	Reset Value	0	0	1	0	0	0	0	0

Bit 7~6 **Reserved**: Reserved bits

Bit 5 **RcvOff**: Receiver analog block on/off control
0: On
1: Off

Bit 4 **PowerDown**: Software power-down mode control
0: NFC reader in ready state
1: NFC reader enters power-down mode

Bit 3~0 **Command[3:0]**: Command control
Based on the value of these bits, the corredsponding command is activated. Reading these bits shows which command is actually being executed. Refer to the Command Overview section for details.

• ComIEn Register

Interrupt enable and disable control bits.

Address	Bit	7	6	5	4	3	2	1	0
02h	Name	IRqInv	TxIEn	RxIEn	IdleIEn	HiAlertIEn	LoAlertIEn	ErrIEn	TimerIEn
	Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	Reset Value	1	0	0	0	0	0	0	0

Bit 7 **IRqInv**: IRQ_NFC pin state non-inverting/inverting with IRq bit state (Status1 register) control
0: Non-invert
1: Invert
In combination with the IRQPushPull bit in the DivIEn register, the default vlaue of logic 1 ensures that the output level on the IRQ_NFC pin is 3-state.

Bit 6 **TxIEn**: Send TxIRq interrupt request to IRQ_NFC pin
0: Not allowed
1: Allowed

Bit 5 **RxIEn**: Send RxIRq interrupt request to IRQ_NFC pin
0: Not allowed
1: Allowed

Bit 4 **IdleIEn**: Send IdleIRq interrupt request to IRQ_NFC pin
0: Not allowed
1: Allowed

Bit 3 **HiAlertIEn**: Send HiAlertIRq interrupt request to IRQ_NFC pin
0: Not allowed
1: Allowed

Bit 2 **LoAlertIEn**: Send LoAlertIRq interrupt request to IRQ_NFC pin
0: Not allowed
1: Allowed

Bit 1 **ErrIEn**: Send ErrIRq interrupt request to IRQ_NFC pin
0: Not allowed
1: Allowed

Bit 0 **TimerIEn**: Send TimerIRq interrupt request to IRQ_NFC pin
0: Not allowed
1: Allowed

• DivIEn Register

Interrupt enable and disable control bits.

Address	Bit	7	6	5	4	3	2	1	0
03h	Name	IRQPushPull	Reserved	TagDetIEn	Reserved		CRCIEn	Reserved	
	Type	R/W	—	R/W	—		R/W	—	
	Reset Value	0	0	0	0	0	0	0	0

Bit 7 **IRQPushPull**: IRQ_NFC output control
0: Open-drain output
1: Standard CMOS output

Bit 6 **Reserved**: Reserved bit

Bit 5 **TagDetIEn**: Send TagDetIRq interrupt request to IRQ_NFC pin
0: Not allowed
1: Allowed

Bit 4~3 **Reserved**: Reserved bits

Bit 2 **CRCIEn**: Send CRCIRq interrupt request to IRQ_NFC pin
0: Not allowed
1: Allowed

Bit 1~0 **Reserved**: Reserved bits

• ComIrq Register

Interrupt request bits. All bits in this register can be cleared by software.

Address	Bit	7	6	5	4	3	2	1	0
04h	Name	Set1	TxIRq	RxIRq	IdleIRq	HiAlertIRq	LoAlertIRq	ErrIRq	TimerIRq
	Type	W	D	D	D	D	D	D	D
	Reset Value	0	0	0	1	0	1	0	0

Bit 7 **Set1**: ComIrq register interrupt request flag control
0: Clear interrupt request flag in the ComIrq register
1: Set interrupt request flag in the ComIrq register

Hardware set high: When a related event occurs, the corresponding interrupt request flag in this register is automatically set high and an interrupt request is generated. It requires to set both Set1 and the corresponding interrupt request flag to 0 to clear the interrupt request.

Software set high: Set both Set1 and the corresponding interrupt request flag in this register to 1 to generate the interrupt request. It requires to set both Set1 and the corresponding interrupt request flag to 0 to clear the interrupt request.

Bit 6 **TxIRq**: Data transmission complete interrupt request flag
0: No request
1: Interrupt request

This flag is set high immediately after the last data bit has been sent.

Bit 5 **RxIRq**: Receiver detected valid data interrupt request flag
0: No request
1: Interrupt request

If the RxNoErr bit in the RxMode register is set high and the data received in the FIFO is valid, this interrupt request flag will be set high.

Bit 4 **IdleIRq**: Idle interrupt request flag
0: No request
1: Interrupt request

If a command terminates, such as changing from any command to the Idle command, this interrupt request flag will be set high. If an unknown command is started, the Command[3:0] field in the Command register will be 0000b and this interrupt request flag will be set high. Starting the Idle command does not set this flag.

Bit 3 **HiAlertIRq**: FIFO HiAlert interrupt request flag
0: No request
1: Interrupt request

This flag will be set high when the HiAlert bit in the Status1 register is set to 1. This flag can only be reset using the Set1 bit of this register.

Bit 2 **LoAlertIRq**: FIFO LoAlert interrupt request flag
0: No request
1: Interrupt request

This flag will be set high when the LoAlert bit in the Status1 register is set to 1. This flag can only be reset using the Set1 bit of this register.

Bit 1 **ErrIRq**: Error interrupt request flag
- 0: No request
- 1: Interrupt request

This flag will be set high when any bit of the Error register is set high.

Bit 0 **TimerIRq**: Timer interrupt request flag
- 0: No request
- 1: Interrupt request

This flag will be set high when the timing value TCounterVal indicated by the TCounterVal_H and TCounterVal_L registers is decreased to 0.

• DivIrq Register

Interrupt request bits. All bits in this register can be cleared by software.

Address	Bit	7	6	5	4	3	2	1	0
05h	Name	Set2	Reserved	TagDetIRq	Reserved		CRCIRq	Reserved	
	Type	W	—	R/W	—		D	—	
	Reset Value	0	0	0	0	0	0	0	0

Bit 7 **Set2**: DivIrq register interrupt request flag control
- 0: Clear interrupt request flag in the DivIrq register
- 1: Set interrupt request flag in the DivIrq register

Hardware set high: When a related event occurs, the corresponding interrupt request flag in this register is automatically set high and an interrupt request is generated. It requires to set both Set2 and the corresponding interrupt request flag to 0 to clear the interrupt request.

Software set high: Set both Set2 and the corresponding interrupt request flag in this register to 1 to generate the interrupt request. It requires to set both Set2 and the corresponding interrupt request flag to 0 to clear the interrupt request.

Bit 6 **Reserved**: Reserved bit

Bit 5 **TagDetIRq**: Card detection interrupt request flag
- 0: No request
- 1: Interrupt request

This flag will be set high when a card has been detected.

Bit 4~3 **Reserved**: Reserved bits

Bit 2 **CRCIRq**: CRC interrupt request flag
- 0: No request
- 1: Interrupt request

This flag will be set high when the CRC command is valid and all data has been checked.

Bit 1~0 **Reserved**: Reserved bits

• Error Register

This register shows the error status of the last command execution.

Address	Bit	7	6	5	4	3	2	1	0
06h	Name	WrErr	TempErr	Reserved	BufferOvfl	CollErr	CRCErr	ParityErr	ProtocolErr
	Type	R	R	—	R	R	R	R	R
	Reset Value	0	0	0	0	0	0	0	0

Bit 7 **WrErr**: Write error flag
- 0: No write error occurs
- 1: Write error occurs

If the host writes data to the FIFO buffer during the MFAuthent command execution or during the period between the last bit transmitted on the RF interface and the last bit received on the RF interface, this flag will be set high.

Bit 6 **TempErr**: Temperature sensor over temperature flag
- 0: No over temperature occurs
- 1: Over temperature occurs

This flag will be set high when the internal temperature sensor has detected an over temperature situation, in which case the antenna drivers will be turned off automatically.

Bit 5 **Reserved**: Reserved bit

Bit 4 **BufferOvfl**: FIFO buffer overflow flag
- 0: Buffer not overflows
- 1: Buffer overflows

When the FIFO buffer is full, if the host or the chip internal state machine (such as receiver) continues to write data to the buffer, this flag will be set high. It can only be cleared by the FlushBuffer bit in the FIFOLevel register.

Bit 3 **CollErr**: Data bit collision flag
- 0: No collision occurs
- 1: Collision occurs

This flag will be set high when a bit collision has been detected. It is automatically cleared during the receiver start-up. This flag is only effective in the bit anti-collision mechanism of the 106kbit/s communication, and is usually set to 0 under 212kbit/s, 424kbit/s and 848kbit/s communication protocols.

Bit 2 **CRCErr**: CRC error flag
- 0: No CRC error occurs
- 1: CRC error occurs

This flag will be set high when the RxCRCEn bit in the RxMode register is set to 1 and a CRC calculation error occurs. It is automatically cleared during the receiver start-up.

Bit 1 **ParityErr**: Parity error flag
- 0: No parity error occurs
- 1: Parity error occurs

This flag will be set high when a parity error occurs. It is automatically cleared during the receiver start-up. This flag is only effective in the 106kbit/s ISO/IEC 14443 A communication.

Bit 0 **ProtocolErr**: SOF(Start of Frame) error flag
- 0: No SOF error occurs
- 1: SOF error occurs

This flag will be set high when the SOF is incorrect. It is automatically cleared during the receiver start-up. This flag is only effective in the 106kbit/s communication.

• Status1 Register

CRC, interrupt and FIFO buffer status bits.

Address	Bit	7	6	5	4	3	2	1	0
07h	Name	Reserved	CRCOk	CRCReady	IRq	TRunning	Reserved	HiAlert	LoAlert
	Type	—	R	R	R	R	—	R	R
	Reset Value	0	0	1	0	0	0	0	1

Bit 7 **Reserved**: Reserved bit

Bit 6 **CRCOk**: CRC calculation correct flag
- 0: CRC calculation is in progress or CRC calculation is incorrect
- 1: CRC calculation is correct

This flag will be set high when the CRC result is 0. Since this bit value is uncertain during transmission or reception, the CRCErr bit in the Error register can be used for CRC result checking. This bit changes to 0 during CRC coprocessor calulation and changes to 1 when the CRC calculation is correctly executed.

Bit 5 **CRCReady**: CRC calculation complete flag
0: CRC calculation is not completed
1: CRC calculation is completed

This flag will be set high when the CRC calculation is completed. It is valid only when executing the CRC calculation of the CalcCRC command.

Bit 4 **IRq**: Interrupt request indication bit
0: No request
1: Interrupt request

To indicate any interrupt request, the corresponding interrupt enable bit must be set. Refer to the ComIEn and DivIEn register.

Bit 3 **TRunning**: Timer running flag
0: Not running
1: Running

When the timer is running, which means when the TCounterVal value is decremented with the timer clock, this flag is set high.

Bit 2 **Reserved**: Reserved bit

Bit 1 **HiAlert**: FIFO high level warning flag

If (64-FIFOLength) ≤ WaterLevel, HiAlert is set to 1.

Bit 0 **LoAlert**: FIFO low level warning flag

If FIFOLength ≤ WaterLevel, LoAlert is set to 1.

• Status2 Register

Transmitter and receiver status bits.

Address	Bit	7	6	5	4	3	2	1	0
08h	Name	TempSensClear	Reserved	Reserved		MFCrypto1On	ModemState[2:0]		
	Type	R/W	—	—		D	R		
	Reset Value	0	0	0	0	0	0	0	0

Bit 7 **TempSensClear**:

When the temperature falls below the 120°C limit alarm value, the temperature error alarm can be removed by setting this bit high.

Bit 6~4 **Reserved**: Reserved bits

Bit 3 **MFCrypto1On**: Crypto_M unit off/on control
0: Off
1: On

Setting this bit to 1 turns on the Crypto_M unit to enter the Crypto_M standard card reader mode, where all data communication with the card is encrypted. This bit can only be set to 1 by a successful execution of the MFAuthent command and is cleared by software.

Bit 5 **ModemState[2:0]**: Transmitter and receiver state machine status
000: Idle
001: Wait for the StartSend bit setting in the BitFraming register
010: TxWait – If the TxWaitRF bit in the Mode register is set to 1, wait until RF field is present. The minimum time for TxWait is determined by TxWait[4:0].
011: Transmitting
100: RxWait – If the TxWaitRF bit in the Mode register is set to 1, wait until RF field is present. The minimum time for RxWait is determined by RxWait[5:0].
101: Wait for data
110: Receiving
111: Undefined

• FIFOData Register

64-byte FIFO buffer input and output.

Address	Bit	7	6	5	4	3	2	1	0
09h	Name	FIFOData[7:0]							
	Type	D							
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~0 **FIFOData[7:0]**: Intput/output port for the internal 64-byte FIFO buffer

It acts as a parallel in / parallel output converter for all data stream inputs and outputs of the buffer.

• FIFOLevel Register

This register indicates the number of bytes stored in the FIFO buffer.

Address	Bit	7	6	5	4	3	2	1	0
0Ah	Name	FlushBuffer	FIFOLevel[6:0]						
	Type	W	R						
	Reset Value	0	0	0	0	0	0	0	0

Bit 7 **FlushBuffer**: Clear FIFO buffer

Setting this bit to 1 will immediately clear the internal FIFO bufrer's read and write pointers, and the BufferOvfl bit in the Error register. Reading this bit always returns 0.

Bit 6~0 **FIFOLevel[6:0]**: Indication for the number of bytes stored in the FIFO buffer

Writing data to the FIFOData register increments the FIFOLevel value and reading data from the FIFOData registesr decrements the FIFOLevel value.

• WaterLevel Register

This register defines the level of FIFO buffer for overflow and empty warning.

Address	Bit	7	6	5	4	3	2	1	0
0Bh	Name	Reserved		WaterLevel[5:0]					
	Type	—		R/W					
	Reset Value	0	0	0	0	1	0	0	0

Bit 7~6 **Reserved**: Reserved bits

Bit 5~0 **WaterLevel[5:0]**: Define the level of FIFO buffer for overflow and underflow warning

If the remaining space in the FIFO buffer is less than or equal to the value defined by WaterLevel[5:0], the HiAlert bit in the Status1 register will be set high.

If the used space in the FIFO buffer is less than or equal to the value defined by WaterLevel[5:0], the LoAlert bit in the Status1 register will be set high.

• Control Register

Various control bits.

Address	Bit	7	6	5	4	3	2	1	0
0Ch	Name	TStopNow	TStartNow	Reserved			RxLastBits[2:0]		
	Type	W	W	—			R		
	Reset Value	0	0	0	1	0	0	0	0

Bit 7 **TStopNow**: Timer immediate stop control

Setting this bit to 1 stops the timer immediately. Reading this bit always returns 0.

Bit 6 **TStartNow**: Timer immediate start control
Setting this bit to 1 starts the timer immediately. Reading this bit always returns 0.

Bit 5~3 **Reserved**: Reserved bits

Bit 2~0 **RxLastBits[2:0]**: Indication for the number of significant bits of last received byte
If this field value is 000b, it means the entire byte is valid.

• BitFraming Register

Adjustment for the bit-oriented frames.

Address	Bit	7	6	5	4	3	2	1	0
0Dh	Name	StartSend	RxAlign[2:0]			Reserved	TxLastBits[2:0]		
	Type	W	R/W			—	R/W		
	Reset Value	0	0	0	0	0	0	0	0

Bit 7 **StartSend**: Data transmission start control
Setting this bit to 1 activates the data transmission. This bit is valid only for the Transceive command execution.

Bit 6~4 **RxAlign[2:0]**: Define the bit position for the first bit received to be stored in the FIFO buffer
000: LSB of the received data is stored at Bit0, the second received bit is stored at Bit1, and so on.
001: LSB of the received data is stored at Bit1, the second received bit is stored at Bit2, and so on.
……
111: LSB of the received data is stored at Bit7, the second received bit is stored at Bit0 of the next byte, and so on.
This bit field is set for bit-oriented frame reception and is only used for the bitwise anti-collision at 106kbit/s. For all other modes this bit field is set to “000”.

Bit 3 **Reserved**: Reserved bit

Bit 2~0 **TxLastBits[2:0]**: Define the number of bits for the last byte to be transmitted
This bit field is set for bit-oriented frame transmission. If this bit field is “000”, it means all bits of the last byte will be transmitted.

• Coll Register

This register defines the first bit collision detected on the RF interface.

Address	Bit	7	6	5	4	3	2	1	0
0Eh	Name	ValuesAfterColl	Reserved	CollPosNotValid	CollPos[4:0]				
	Type	R/W	—	R	R				
	Reset Value	1	0	1	0	0	0	0	0

Bit 7 **ValuesAfterColl**:
If this bit is set to 0, all data bits will be cleared after a collision. It is only used in bitwise anti-collision at 106kbit/s, otherwise it is set to 1.

Bit 6 **Reserved**: Reserved bit

Bit 5 **CollPosNotValid**:
If this bit is set to 1, it indicates that no collision is detected or the collision bit position is outside the rang defined by CollPos[4:0].

Bit 4~0 **CollPos[4:0]**: Position of the first collistion bit detected in the received frame (only data bits are interpreted)
00000: Indicates a bit collision at the 32nd bit
00001: Indicates a bit collision at the 1st bit
……

01000: Indicates a bit collision at the 8th bit
……

These bits are only interpreted when the CollPosNotValid bit is 0.

Page 1

• Mode Register

This register defines the general mode settings for transmission and reception.

Address	Bit	7	6	5	4	3	2	1	0
11h	Name	MSBFirst	Reserved	TxWaitRF	Reserved			CRCPreset[1:0]	
	Type	R/W	—	R/W	—			R/W	
	Reset Value	0	0	1	1	1	1	0	1

Bit 7 **MSBFirst**:
When this bit is set to 1, the CRC coprocessor will calculate the CRC with the MSB first. The calculation result is stored in the CRCResult_H and CRCResult_L registers. During RF communication, this bit is ignored.

Bit 6 **Reserved**: Reserved bit

Bit 5 **TxWaitRF**:
If this bit is set to 1, the transmitter is started when the RF field presents.

Bit 4~2 **Reserved**: Reserved bits

Bit 1~0 **CRCPreset[1:0]**: Define the preset value for the CRC coprocessor to execute CalcCRC command
00: 0000h
01: 6363h
10: A671h
11: FFFFh

During any communication, the preset value is selected automatically according to the bit definition in the RxMode and TxMode registers.

• TxMode Register

This register defines the data rate during transmission.

Address	Bit	7	6	5	4	3	2	1	0
12h	Name	TxCRCEn	TxSpeed[2:0]			InvMod	Reserved	TxFraming[1:0]	
	Type	R/W	D			R/W	—	D	
	Reset Value	0	0	0	0	0	0	0	0

Bit 7 **TxCRCEn**:
Setting this bit to 1 enables the CRC generation during the data transmission.

Bit 6~4 **TxSpeed[2:0]**: Define the bit rate during the data transmission
000: 106kbit/s
001: 212kbit/s
010: 424kbit/s
011: 848kbit/s
100~111: Reserved

This chip supports transmission rates up to 848kbit/s.

Bit 3 **InvMod**:
If this bit is set to 1, the inverting value of the modulated data is sent.

Bit 2 **Reserved**: Reserved bit

Bit 1~0 **TxFraming[1:0]**: Define the transmission data structure
00: ISO/IEC 14443 A / Crypto_M
01~10: Reserved
11: ISO/IEC 14443 B

• RxMode Register

This register defines the data rate during the data reception.

Address	Bit	7	6	5	4	3	2	1	0
13h	Name	RxCRCEn	RxSpeed[2:0]			RxNoErr	RxMultiple	RxFraming	
	Type	R/W	D			R/W	R/W	D	
	Reset Value	0	0	0	0	0	0	0	0

Bit 7 **RxCRCEn**:

Setting this bit to 1 enables the CRC generation during the data reception.

Bit 6~4 **RxSpeed[2:0]**: Define the bit rate during the data reception

000: 106kbit/s
001: 212kbit/s
010: 424kbit/s
011: 848kbit/s
100~111: Reserved

This chip supports transmission rates up to 848kbit/s.

Bit 3 **RxNoErr**:

If this bit is set to 1, an invalid data stream received (less than 4 bits) will be ignored and the receiver remains active.

Bit 2 **RxMultiple**:

0: Receiver stops after receiving one data frame
1: Receiver continuously receives multiple data frames

After this bit is set to 1, the Receive and Transceive commands will not terminate automatically. Continuous reception can be stopped by writing any command code to the Command register (except Receive command) or by clearing the bit by the host.

Bit 1~0 **RxFraming[1:0]**: Define the data structure to be received

00: ISO/IEC 14443 A / Crypto_M
01~10: Reserved
11: ISO/IEC 14443 B

• TxControl Register

This register controls the logic status of the antenna driver pins TX1A and TX2A.

Address	Bit	7	6	5	4	3	2	1	0
14h	Name	Reserved	InvTxRFOn	Reserved		TxCW	Reserved	TxRFEn	
	Type	—	R/W	—		R/W	—	R/W	
	Reset Value	1	0	0	0	0	0	0	0

Bit 7 **Reserved**: Reserved bit

Bit 6 **InvTxRFOn**:

If this bit is set to 1, the carrier on the TX1A and TX2A pins are inverted when driver is enabled.

Bit 5~4 **Reserved**: Reserved bits

Bit 3 **TxCW**:

0: Do not output the unmodulated 13.56MHz energy carrier
1: Continuously output the unmodulated 13.56MHz energy carrier

Bit 2 **Reserved**: Reserved bit

Bit 1~0 **TxRFEn**:

00: Do not output the modulated 13.56MHz energy carrier
01: Continuously output the modulated 13.56MHz energy carrier on TX1A
10: Continuously output the modulated 13.56MHz energy carrier on TX2A
11: Continuously output the modulated 13.56MHz energy carrier on both TX1A and TX2A

• TxASK Register

This register controls the transmitting modulation.

Address	Bit	7	6	5	4	3	2	1	0
15h	Name	Reserved	Force100ASK	Reserved					
	Type	—	R/W	—					
	Reset Value	0	1	0	0	0	0	0	0

Bit 7 **Reserved**: Reserved bit

Bit 6 **Force100ASK**:

Setting this bit to 1 will enforce a 100% ASK modulation independent of the ModGsP register setting.

Bit 5~0 **Reserved**: Reserved bits

• TxSel Register

This register selects the signal source for the antenna drivers.

Address	Bit	7	6	5	4	3	2	1	0
16h	Name	Reserved		DriverSel[1:0]		Reserved			
	Type	—		R/W		—			
	Reset Value	0	0	0	1	0	0	0	0

Bit 7~6 **Reserved**: Reserved bits

Bit 5~4 **DriverSel[1:0]**: Select the input signal of drivers TX1A and TX2A
00: 3-state – If selected, the drivers can only be in 3-state mode in the software power-down mode
01: Modulation signal (envolope) from the internal encoder, Miller encoded
10: Reserved bits
11: High level – depends on the InvTxRFOn bit setting

Bit 3~0 Reserved: Reserved bits

• RxSel Register

This registers controls the internal receiver settings.

Address	Bit	7	6	5	4	3	2	1	0
17h	Name	UARTSel[1:0]		RxWait[5:0]					
	Type	R/W		R/W					
	Reset Value	1	0	0	0	1	0	0	0

Bit 7~6 **UARTSel[1:0]**: Select contactless UART input
00: Low level
01: Reserved bits
10: Modulated signal from the internal analog block
11: Reserved bits

Bit 5~0 **RxWait[5:0]**: Define the delay time before activating the receiver

After data transmission, the activation of the receiver has a delay of RxWait[5:0]×ETU, where ETU is defined by the RxWaitEtu bit in the Special register. During this frame guard time, any signal on the RXA pin will be ignored. This parameter is ignored by the Receive command and is used by all other commnads. The counter starts immediately after the external RF field is switched on.

• Demod Register

This register defines the demodulation settings.

Address	Bit	7	6	5	4	3	2	1	0
19h	Name	AddIQ[1:0]		FixIQ	TPrescalEven	Reserved			
	Type	R/W		R/W	R/W	—			
	Reset Value	0	1	0	0	1	1	0	1

Bit 7~6 **AddIQ[1:0]**: Define the use of I and Q channels during reception
00: Select a stronger signal channel
01: Select a stronger signal channel and freeze the selected channel during communication
10~11: Reserved
Note that the FixIQ bit must be set to 0 to enable the above settings.

Bit 5 **FixIQ**:
When this bit is set to 1, if AddIQ[1:0] is set to "x0", the reception is fixed to I channel; if AddIQ[1:0] is set to "x1", the reception is fixed to Q channel.

Bit 4 **TPrescalEven**: Select the formula to calculate the timer frequency
0: f_{Timer}=13.56MHz/(2×TPrescaler+1)
1: f_{Timer}=13.56MHz/(2×TPrescaler+2)
By default, the formula of option 0 is used for calculation. Refer to the Tmode and TPrescaler registers for TPrescaler definition.

Bit 3~0 **Reserved**: Reserved bits

• MfTx Register

This registesr defines the transmission waiting time.

Address	Bit	7	6	5	4	3	2	1	0
1Ch	Name	Reserved			TxWait[4:0]				
	Type	—			R/W				
	Reset Value	0	1	1	0	0	0	1	0

Bit 7~5 **Reserved**: Reserved bits

Bit 4~0 **TxWait[4:0]**: Define the transmission waiting time
By default, waiting time=(TxWait[4:0])×ETU, where ETU is defined by the TxWaitEtu bit in the Special register.

• MfRx Register

Address	Bit	7	6	5	4	3	2	1	0
1Dh	Name	Reserved			ParityDisable	Reserved		RxHPF[1:0]	
	Type	—			R/W	—		R/W	
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~5 **Reserved**: Reserved bits

Bit 4 **ParityDisable**: Transmission and reception parity function on/off control
0: On
1: Off
When this bit is set to 1, the parity bit generation for transmission and parity check for reception are turned off, and the received parity bit is processed as data bit.

Bit 3~2 **Reserved**: Reserved bits

Bit 1~0 **RxHPF[1:0]**: Select high-pass bandwidth
00: 72kHz
01: 100kHz
10: 150kHz
11: 300kHz
Set to “11” at 106kbit/s, “10” at 212kbit/s, “01” at 424kbit/s and “00” at 848kbit/s.

• TypeB Register

This register configures the ISO/IEC 14443 B functions.

Address	Bit	7	6	5	4	3	2	1	0
1Eh	Name	RxSOFReq	RxEOFReq	Reserved	EOFSOFWidth	NoTxSOF	NoTxEOF	TxEGT[1:0]	
	Type	R/W	R/W	—	R/W	R/W	R/W	R/W	
	Reset Value	0	0	0	0	0	0	0	0

Bit 7 **RxSOFReq**:
0: Receive data streams with or without SOF; remove SOF and do not write them to FIFO
1: Require SOF; ignore data streams without SOF

Bit 6 **RxEOFReq**:
0: Receive data streams with or without EOF; remove EOF and do not write them to FIFO
1: Require EOF; data stream without EOF generates a protocol error

Bit 5 **Reserved**: Reserved bit

Bit 4 **EOFSOFWidth**:
0: Define the minimum length of SOF and EOF in IOS/IEC 14443 B
1: Define the maximum length of SOF and EOF in IOS/IEC 14443 B

Bit 3 **NoTxSOF**:
If this bit is set to 1, the SOF will be omitted from the transmitted framing.

Bit 2 **NoTxEOF**:
If this bit is set to 1, the EOF will be omitted from the transmitted framing.

Bit 1~0 **TxEG[1:0]**: Define EGT(Extra Guard Time) length
00: No EGT
01: 1 bit
10: 2 bits
11: 3 bits

• SerialSpeed Register

This register selects the UART data rate. Refer to the UART Interface section for detailed configuration.

Address	Bit	7	6	5	4	3	2	1	0
1Fh	Name	BR_T0[2:0]			BR_T1[4:0]				
	Type	R/W			R/W				
	Reset Value	1	1	1	0	1	0	1	1

Bit 7~5 **BR_T0[2:0]**: Adjust transmission data rate

Bit 4~0 **BR_T1[4:0]**: Adjust transmission data rate

Page 2

• CRCResult_H Register

This register shows the high byte value of the CRC calculation result.

Address	Bit	7	6	5	4	3	2	1	0
21h	Name	CRCResultMSB[7:0]							
	Type	R							
	Reset Value	1	1	1	1	1	1	1	1

Bit 7~0 **CRCResultMSB[7:0]**: CRC calculation result high byte value
It is valid only when the CRCReady bit in the Status1 register is set high.

• CRCResult_L Register

This register shows the low byte value of the CRC calculation result.

Address	Bit	7	6	5	4	3	2	1	0
22h	Name	CRCResultLSB[7:0]							
	Type	R							
	Reset Value	1	1	1	1	1	1	1	1

Bit 7~0 **CRCResultLSB[7:0]**: CRC calculation result low byte value
It is valid only when the CRCReady bit in the Status1 register is set high.

• ModWidth Register

This register sets the modulation width.

Address	Bit	7	6	5	4	3	2	1	0
24h	Name	ModWidth[7:0]							
	Type	R/W							
	Reset Value	0	0	1	0	0	1	1	0

Bit 7~0 **ModWidth[7:0]**: Define the Miller modulation width to be (ModWidth+1) times the carrier frequency
The maximum value is half a bit cycle.

• RFCfg Register

This register configures the receiver gain.

Address	Bit	7	6	5	4	3	2	1	0
26h	Name	Reserved	RxGain[3:0]				Reserved		
	Type	—	R/W				—		
	Reset Value	0	1	0	0	1	0	0	0

Bit 7 **Reserved**: Reserved bit

Bit 6~3 **RxGain[3:0]**: Define the receiver signal voltage gain factor

0000	18dB	1000	34dB
0001	20dB	1001	36dB
0010	22dB	1010	38dB
0011	24dB	1011	40dB
0100	26dB	1100	42dB
0101	28dB	1101	44dB
0110	30dB	1110	46dB
0111	32dB	1111	48dB

Bit 2~0 **Reserved**: Reserved bits

- **GsN Register**

This register defines the conductance value when antenna drivers TX1A and TX2A are turned on and used as N drivers.

Address	Bit	7	6	5	4	3	2	1	0
27h	Name	CWGsN[3:0]				ModGsN[3:0]			
	Type	R/W				R/W			
	Reset Value	1	0	0	0	1	0	0	0

Bit 7~4 **CWGsN[3:0]**: Define the conductance value of N driver when it outputs a CW signal

The conductance value is binary-weighted. The untit conductance value of the N driver is 1/160 S. By setting this register, the conductance value of the N driver during the non-modulation period is: CWGsN[3:0]×(1/160) S.

This field setting is valid only when the drivers TX1A and TX2A are turned on.

The highest bit is forced to 1 in the software power-down mode.

Bit 3~0 **ModGsN[3:0]**: Define the conductance value of N driver when it outputs a MOD signal

The conductance value is binary-weighted. The untit conductance value of the N driver is 1/160 S. By setting this register, the conductance value of the N driver during the modulation period is: ModGsN[3:0]×(1/160) S.

This field setting is valid only when the drivers TX1A and TX2A are turned on.

The highest bit is forced to 1 in the software power-down mode.

- **CWGsP Register**

This register defines the conductance value of the P driver during the non-modulation period.

Address	Bit	7	6	5	4	3	2	1	0
28h	Name	Reserved		CWGsP[5:0]					
	Type	—		R/W					
	Reset Value	0	0	1	0	0	0	0	0

Bit 7~6 **Reserved**: Reserved bits

Bit 5~0 **CWGsP[5:0]**: Define the conductance value of P driver when it outputs a CW signal

The conductance value is binary-weighted. The untit conductance value of the P driver is 1/640 S. By setting this register, the conductance value of the P driver during the non-modulation period is: CWGsP[5:0]×(1/640) S.

The highest bit is forced to 1 in the software power-down mode.

- **ModGsP Register**

This register defines the conductance value of the P driver during the modulation period.

Address	Bit	7	6	5	4	3	2	1	0
29h	Name	Reserved		ModGsP[5:0]					
	Type	—		R/W					
	Reset Value	0	0	1	0	0	0	0	0

Bit 7~6 **Reserved**: Reserved bits

Bit 5~0 **ModGsP[5:0]**: Define the conductance value of P driver when it outputs a MOD signal

The conductance value is binary-weighted. The untit conductance value of the P driver is 1/640 S. By setting this register, the conductance value of the P driver during the modulation period is: ModGsP[5:0]×(1/640) S.

Even if the Force100ASK bit in the TxASK register is set to 1, it has no effect on the ModGsP[5:0] value.

The highest bit is forced to 1 in the software power-down mode.

• TMode Register

This register defines the timer settings.

Address	Bit	7	6	5	4	3	2	1	0
2Ah	Name	TAuto	Reserved		TAutoRestart	TPrescaler_Hi[3:0]			
	Type	R/W	R/W		R/W	R/W			
	Reset Value	1	0	0	0	1	0	0	0

Bit 7 **TAuto**: Timer auto start control
- 0: Timer is not affected by this cotrol bit
- 1: Timer starts automatically at the end of data transmission at all communication speeds

After this bit is set to 1, if the RxMultiple bit in the RxMode register is 0, the timer will stop running immediately after receiving the 5th bit (1 start bit and 4 data bits); if the RxMultiple bit is 1 the timer will not stop, in which case the timer can only be stopped by setting the TstopNow bit in the Control register to 1.

Bit 6~5 **Reserved**: Reserved bits

Bit 4 **TAutoRestart**:
- 0: Timer decrements to 0 and the TimerIRq flag in the ComIrq register is set to 1
- 1: Timer automatically restarts down-counting from the 16-bit timer reload value

Bit 3~0 **TPrescaler_Hi[3:0]**: TPrescaler value higher 4 bits

Refer to the TPrescaler register.

• TPrescaler Register

This register defines the timer settings.

Address	Bit	7	6	5	4	3	2	1	0
2Bh	Name	TPrescaler_Lo[7:0]							
	Type	R/W							
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~0 **TPrescaler_Lo[7:0]**: TPrescaler value lower 8 bits

If the TPrescalEven bit in the Demod registrer is 0: f_{Timer}=13.56MHz/(2×TPrescaler+1)

If the TPrescalEven bit in the Demod registrer is 1: f_{Timer}=13.56MHz/(2×TPrescaler+2)

TPrescaler=[TPrescaler_Hi:TPrescaler_Lo]

• TReload_H Register

This register defines the high byte of the 16-bit timer reload value.

Address	Bit	7	6	5	4	3	2	1	0
2Ch	Name	TReloadVal_Hi[7:0]							
	Type	R/W							
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~0 **TReloadVal_Hi[7:0]**: Timer 16-bit reload value higher 8 bits

When a start event occurs, the 16-bit reload value is loaded into the timer. Changing this register will only affect the timer on the next start event.

• TReload_L Register

This register defines the low byte of the 16-bit timer reload value.

Address	Bit	7	6	5	4	3	2	1	0
2Dh	Name	TReloadVal_Lo[7:0]							
	Type	R/W							
	Reset Value	0	0	0	1	0	0	0	0

Bit 7~0 **TReloadVal_Lo[7:0]** : Timer 16-bit reload value lower 8 bits
When a start event occurs, the 16-bit reload value is loaded into the timer. Changing this register will only affect the timer on the next start event.

• TCounterVal_H Register

This register shows the high byte of the timer current value.

Address	Bit	7	6	5	4	3	2	1	0
2Eh	Name	TCounterVal_Hi[7:0]							
	Type	R							
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~0 **TCounterVal_Hi[7:0]**: Timer current value higher 8 bits

• TCounterVal_L Register

This register shows the low byte of the timer current value.

Address	Bit	7	6	5	4	3	2	1	0
2Fh	Name	TCounterVal_Lo[7:0]							
	Type	R							
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~0 **TCounterVal_Lo[7:0]**: Timer current value lower 8 bits

Page 3

• PageSel Register

Address	Bit	7	6	5	4	3	2	1	0
37h	Name	PageSel[7:0]							
	Type	R/W							
	Reset Value	0	0	0	1	0	1	0	1

Bit 7~0 **PageSel[7:0]**: Private register switch control
0x5E: Enable access to Page 4
0xAE: Enable access to Page 5
0x5A: Enable access to Page 6
To configure the registers in pages 4~6, this register should first be configured to the correct switch value before executing any operations to the registers in the corresponding page.

Private Register Description

Page 4

PageSel[7:0] must first be set to 0x5E so as to configure the registers in page 4.

• RxAlgorithm0 Register

Address	Bit	7	6	5	4	3	2	1	0
33h	Name	Reserved						BSAdjCnt	BSAdjDis
	Type	R						R/W	R/W
	Reset Value	0	0	0	0	0	0	0	1

Bit 7~2 **Reserved**: Reserved bits

Bit 1 **BSAdjCnt**: Select the number of bit cycles used for bit synchronization of the OOK demodulation algorithm
- 0: 4 bit cycles
- 1: 8 bit cycles

Bit 0 **BSAdjDis**: Control the bit synchronization of the OOK demodulation altorithm
- 0: Enable
- 1: Disable

• AGCCfg0 Register

Address	Bit	7	6	5	4	3	2	1	0
34h	Name	Reserved							First_Gain_Indx[2]
	Type	R							R/W
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~1 **Reserved**: Reserved bits

Bit 0 **First_Gain_Indx[2]**: AGC (Automatic Gain Control) first step gain control bit 2

First_Gain_Indx[1:0] is located in the AGCCfg1 register.

• AGCCfg1 Register

Address	Bit	7	6	5	4	3	2	1	0
35h	Name	EXT_AGC_en		Full_Scale_Num[1:0]		First_Gain_Indx[1:0]		Gain_Step[1:0]	
	Type	R/W		RR/W		RR/W		RR/W	
	Reset Value	1	1	1	0	1	1	0	0

Bit 7~6 **EXT_AGC_en**: AGC function control
- 00: Disable AGC
- 01: Enable – signal stage
- 10: Enable – RxWait stage, signal stage
- 11: Enable – RxWait stage, post-RxWait signal stage, signal stage

Bit 5~4 **Full_Scale_Num[1:0]**: Define the number of saturation points for gain reduction in the automatic gain mode

Bit 3~2 **First_Gain_Indx[1:0]**: AGC first step gain control bit 1~0

First_Gain_Indx[2] is located in the AGCCfg0 register.

First_Gain_Indx[2:0]=
- 000: 48dB
- 001: 44dB
- 010: 40dB
- 011: 36dB
- 100: 32dB
- 101: 42dB

110: 38dB
111: 34dB

Bit 1~0 **Gain_Step[1:0]**: Select the gain decrement in the automatic gain mode
00: 2dB
01: 4dB
10: 6dB
11: Reserved

• RxAlgorithm1 Register

Address	Bit	7	6	5	4	3	2	1	0
36h	Name	IQFixEn	ManRxLPF	RxLPF[1:0]		EnRx2Bit	DC_bp	Reserved	
	Type	RW	RW	RW		RW	RW	RW	
	Reset Value	0	0	0	0	0	0	0	0

Bit 7 **IQFixEn**: Control the IQ channel manual selection in the Demod register
0: Disable
1: Enable

Bit 6 **ManRxLPF**: Enable the low-pass bandwidth manual selection
0: Automatic selection
1: Manual selection

Bit 5~4 **RxLPF[1:0]**: Low-pass bandwidth selection
00: 1.6MHz
01: 1.8MHz
10: 2.0MHz
11: 2.3MHz

Set to “00” at 106kbit/s, set to “01” at 212kbit/s, set to “10” at 424kbit/s and set to “11” at 848kbit/s.

Bit 3 **EnRx2Bit**: Improve noise immunity by algorithm
0: Enable – In Type A, the first 2 significant data bits are used to improve noise immunity, so data shorter than 2 bits cannot be received
1: Disable

Bit 2 **DC_bp**: Demodulation altorithm DC removal control
0: Enable DC removal
1: Disable DC removal

Bit 1~0 **Reserved**: Reserved bits

• RxAlgorithm2 Register

Address	Bit	7	6	5	4	3	2	1	0
38h	Name	MinLevel[3:0]				CollRatio[3:0]			
	Type	RW				RW			
	Reset Value	0	1	1	0	1	0	1	1

Bit 7~4 **MinLevel[3:0]**: Demodulation algorithm minimum signal detection amplitude paramter 2

Bit 3~0 **CollRatio[3:0]**: Demodulation algorithm TypeA collision judgement criteria parameter

- **RxAlgorithm3 Register**

Address	Bit	7	6	5	4	3	2	1	0
39h	Name	Reserved	TALevel[2:0]			EnergyLevel[3:0]			
	Type	R	RW			RW			
	Reset Value	0	0	1	0	1	0	0	0

Bit 7 **Reserved**: Reserved bit

Bit 6~4 **TALevel[2:0]**: Demodulation algorithm transmission end criteria parameter 1

Bit 3~0 **EnergyLevel[3:0]**: Demodulation algorithm transmission end criteria parameter 2

- **RxCK Register**

Address	Bit	7	6	5	4	3	2	1	0
3Ah	Name	Reserved		Reserved	TypeADT	CKDlyAuto	CKDlySel[2:0]		
	Type	RW		RW	RW	RW	RW		
	Reset Value	0	0	0	0	1	0	0	0

Bit 7~5 **Reserved**: Reserved bits

Bit 4 **TypeADT**: TypeA wave falling time adjustment
0: The function does not take effect
1: Speed up the Type A wave falling edge

Bit 3 **CKDlyAuto**: IQ automatic phase selection
0: IQ phase is specified by CKDlySel[2:0]
1: IQ phase is automatically locked

Bit 2~0 **CKDlySel[2:0]**:
There are eight levels by setting this field from 000 to 111. It is used to select the phase angle when configuring the internal signal demodulation, which affects the card reading effect.

- **RxBand Register**

Address	Bit	7	6	5	4	3	2	1	0
3Bh	Name	Reserved		MchAckH[2:0]			MchAckL[2:0]		
	Type	—		R/W			R/W		
	Reset Value	0	0	1	0	0	1	0	1

Bit 7~6 **Reserved**: Reserved bits

Bit 5~3 **MchAckH[2:0]**: Manchester encoding SOF detection, half Bit1 / Noise threshold factor

Bit 2~0 **MchAckL[2:0]**: Manchester encoding SOF detection, half Bit0 / half Bit1 threshold factor

- **LPCD Register**

Address	Bit	7	6	5	4	3	2	1	0
3Ch	Name	Reserved	Reserved	CLK32K_En	CalibEn	Delta[3:0]			
	Type	R	—	R/W	R/W	R/W			
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~6 **Reserved**: Reserved bits

Bit 5 **CLK32K_En**: 32kHz clock enable
0: Disable
1: Enable

Bit 4	**CalibEn**: Card detection calibration enable 0: Disable 1: Enable
Bit 3~0	**Delta[3:0]**: Define the difference between the detected values with and without card presence

- **WUPeriod Register**

Address	Bit	7	6	5	4	3	2	1	0
3Dh	Name	WUPeriod[7:0]							
	Type	R/W							
	Reset Value	0	0	0	0	1	1	1	1

Bit 7~4	**WUPeriod[7:0]**: Sleep time setting T[inactivity]=WUPeriod×256×Tclk_32k

- **SwingsCnt Register**

Address	Bit	7	6	5	4	3	2	1	0
3Eh	Name	LPCD_en	Skip[2:0]			SwingsCnt[3:0]			
	Type	R/W	R/W			R/W			
	Reset Value	0	0	0	0	0	1	0	0

Bit 7	**LPCD_en**: Card detection enable in software reset 0: Disable 1: Enable
Bit 6~4	**Skip[2:0]**: Enhance card detection stability and prevent false wake-up For example, when the tag changes position, the initial detection wave may become unstable, or the detection wave may be unstable at high and low temperatures, which may cause false wake-up. Setting Skip[2:0] can prevent false wake-ups caused by signal jitter. When the number of card detections exceeds the value defined by this field, a TagDetIRq interrupt request will be generated.
Bit 3~0	**SwingsCnt[3:0]**: Detection time setting T[detect]=SwingsCnt×16×4×Tclk_27M12

- **Special Register**

Address	Bit	7	6	5	4	3	2	1	0
3Fh	Name	ThrAck[1:0]		BPSKBSMode		Reserved		RxWaitEtu	TxWaitEtu
	Type	R/W		R/W		R/W		R/W	R/W
	Reset Value	0	1	0	1	0	0	0	0

Bit 7~6	**ThrAck[1:0]**: Demodulation algorithm minimum signal detection amplitude paramter 3
Bit 5~4	**BPSKBSMode[1:0]**: Demodulation algorithm selection 00: Auto 01: Algorithm 1 10: Algorithm 2 11: Undefined
Bit 3~2	**Reserved**: Reserved bits
Bit 1	**RxWaitEtu**: RxWait ETU (Elementary Time Unit) setting 0: RxWait[5:0] bits in the RxSel register take effect, ETU=128/13.56MHz (about 9.4µs) 1: RxWait[5:0] bits in the RxSel register take effect, ETU=64/13.56MHz (about 4.7µs)

Bit 0 **TxWaitEtu**: TxWait ETU (Elementary Time Unit) setting
0: TxWait[4:0] bits in the MfTx register take effect, ETU=128/13.56MHz (about 9.4µs)
1: TxWait[4:0] bits in the MfTx register take effect, ETU=64/13.56MHz (about 4.7µs)

Page 5

PageSel[7:0] must first be set to 0xAE so as to configure the registers in page 5.

• Analog Register

Address	Bit	7	6	5	4	3	2	1	0
31h	Name	Reserved		TEMP_en	TEMP_Protect[1:0]		RefCtrl[1:0]		Reserved
	Type	—		R/W	R/W		R/W		—
	Reset Value	0	0	0	1	0	1	0	0

Bit 7~6 **Reserved**: Reserved bits

Bit 5 **TEMP_en**: High temperature protection enable
0: On
1: Off

Bit 4~3 **TEMP_Protect[1:0]**: High temperature protection threshold selection
00: Undefined
01: 130°C
10: 140°C (default)
11: 150°C

Bit 2~1 **RefCtrl[1:0]**: ADC quantifiable range control
00: 440mV
01: 520mV
10: 600mV
11: 680mV

Bit 0 **Reserved**: Reserved bit

• Noise Register

Address	Bit	7	6	5	4	3	2	1	0
32h	Name	NoiseEstm [2:0]			ForceReSync_BPSK	BPSKEndFactor[1:0]		OOKEndFactor[1:0]	
	Type	R/W			R/W	R/W		R/W	
	Reset Value	0	1	1	1	1	1	0	1

Bit 7~5 **NoiseEstm[2:0]**: Noise evaluation time length factor

Bit 4 **ForceReSync_BPSK**:
If this bit is set to 1, execute resynchronization after demodulating each character of the BPSK frame.

Bit 3~2 **BPSKEndFactor[1:0]**: BPSK modulation end energy threshold factor

Bit 1~0 **OOKEndFactor[1:0]**: OOK modulation end energy threshold factor

• StepCtrl Register

Address	Bit	7	6	5	4	3	2	1	0
33h	Name	Reserved				StepCtrln	StepCtrlp	Reserved	
	Type	R				R/W	R/W	R	
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~4 **Reserved**: Reserved bits

Bit 3 **StepCtrln**: Transmission modulation N driver control bit to control the rising and falling edges of the A/B waveforms

Bit 2	**StepCtrlp**: Transmission modulation P driver control bit to control the rising and falling edges of the A/B waveforms
Bit 1~0	**Reserved**: Reserved bits

• AgcMin Register

Address	Bit	7	6	5	4	3	2	1	0
34h	Name	Reserved		AgcMinValue3			AgcMinValue2		
	Type	R		R/W			R/W		
	Reset Value	0	0	0	1	0	0	0	0

Bit 7~6	**Reserved**: Reserved bits
Bit 5~3	**AgcMinValue3**: AGC amplitude threshold 3
Bit 2~0	**AgcMinValue2**: AGC amplitude threshold 2

• RxAlgorithm6 Register

Address	Bit	7	6	5	4	3	2	1	0
38h	Name	Dis848kFlt	PeakValFactor[2:0]			DisSubEndChk	NoiseLimt[2:0]		
	Type	R/W	R/W			R/W	R/W		
	Reset Value	0	1	1	1	0	1	1	0

Bit 7	**Dis848kFlt**: 848kbit/s subcarrier passing through filter control bit 0: Enable 1: Disable
Bit 6~4	**PeakValFactor[2:0]**: Subcarrier detection mean amplitude threshold factor
Bit 3	**DisSubEndChk**: Subcarrier end detection control bit 0: Enable 1: Disable
Bit 2~0	**NoiseLimt[2:0]**: Noise detection threshold

• RxAlgorithm7 Register

Address	Bit	7	6	5	4	3	2	1	0
39h	Name	CntConfig[1:0]		PulseFactor[1:0]		WidthLimt[1:0]		OffsetLimt[1:0]	
	Type	R/W		R/W		R/W		R/W	
	Reset Value	0	1	0	1	0	1	1	0

Bit 7~6	**CntConfig[1:0]**: Subcarrier detection cycle threshold factor 2
Bit 5~4	**PulseFactor[1:0]**: Subcarrier detection energy threshold factor 2
Bit 3~2	**WidthLimt[1:0]**: Subcarrier detection cycle threshold factor 3
Bit 1~0	**OffsetLimt[1:0]**: Subcarrier detection cycle threshold factor 4

• RxAlgorithm8 Register

Address	Bit	7	6	5	4	3	2	1	0
3Ah	Name	MinPeakVal			StartBit0Limt[2:0]			StartBit1Limt[1:0]	
	Type	R/W			R/W			R/W	
	Reset Value	0	0	1	1	0	0	0	1

Bit 7~5	**MinPeakVal**: Subcarrier detection minimum amplitude threshold
Bit 4~2	**StartBit0Limt[2:0]**: Half-Bit0 detection cycle threshold for Manchester frame start bit detection
Bit 1~0	**StartBit1Limt[1:0]**: Half-Bit1 detection cycle threshold for Manchester frame start bit detection

• RxAlgorithm9 Register

Address	Bit	7	6	5	4	3	2	1	0
3Bh	Name	AgcMinValue1			AgcCntNum			PreambleLimt[1:0]	
	Type	RW			R/W			R/W	
	Reset Value	0	0	0	1	0	0	1	0

Bit 7~5 **AgcMinValue1**: AGC amplitude threshold 1

Bit 4~2 **AgcCntNum**: AGC cycle threshold

Bit 1~0 **PreambleLimt[1:0]**: Preamble detection threshold

Page 6

PageSel[7:0] must first be set to 0x5A so as to configure the registers in page 6.

• LPCDRef Register

Address	Bit	7	6	5	4	3	2	1	0
31h	Name	ReferenceValue[7:0]							
	Type	R							
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~0 **ReferenceValue[7:0]**: LPCD detection reference value

Note that when there is no tag (i.e., no load) above the antenna, adjust this value to differ from the LPCDADCRef register setting by ±5.

• LPCDDet Register

Address	Bit	7	6	5	4	3	2	1	0
32h	Name	DetectedValue[7:0]							
	Type	R							
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~0 **DetectedValue[7:0]**: LPCD detection detected value

• Calibration Register

Address	Bit	7	6	5	4	3	2	1	0
33h	Name	CalibStep		CalibMode	LPCDADCManEn	LPCDEnRCcal	RC32KCalMan	RC27MCalMan	LPCDUseRC
	Type	R/W		R/W	R/W	R/W	R/W	R/W	R/W
	Reset Value	0	0	0	0	0	0	0	0

Bit 7~6 **CalibStep**: CWGsP_lpcd adjustment step size and range when CalibMode=1
- 00: Step=1, [CWGsP_lpcd-8, CWGsP_lpcd+7]
- 01: Step=2, [CWGsP_lpcd-16, CWGsP_lpcd+14]
- 10: Step=3, [CWGsP_lpcd-24, CWGsP_lpcd+21]
- 11: Step=4, [2, 62]

Bit 5 **CalibMode**: When CalibEn=1 and the LPCD mode is entered
- 0: Directly use the ADC sample data as the reference value
- 1: Adjust CWGsP_lpcd, use the ADC data close to LPCDADCRef setting as the reference value

Bit 4 **LPCDADCManEn**: LPCD ADC manual enable

Bit 3 **LPCDEnRCcal**: Enable RC27MHz and RC32kHz calibration after the end of LPCD

When LPCD_en=1, enforce the RC27MHz calibration once;

When LPCD_en=1 or CLK32K_En=1, enforce the RC32kHz calibration once.

Bit 2 **RC32KCalMan**: LPCD 32kHz RC manual calibration enable

Bit 1 **RC27MCalMan**: LPCD 27.12MHz RC manual calibration enable

Bit 0 **LPCDUseRC**: LPCD 27.12MHz RC enable

• RC27MCalValue Register

Address	Bit	7	6	5	4	3	2	1	0
34h	Name	RC27MCalValue							
	Type	R/W							
	Reset Value	1	0	0	0	0	0	0	0

Bit 7~0 **RC27MCalValue**: LPCD 27.12MHz RC calibration value

• RC32KCalValue Register

Address	Bit	7	6	5	4	3	2	1	0
35h	Name	Reserved		RC32KCalValue					
	Type	—		R/W					
	Reset Value	0	0	1	0	0	0	0	0

Bit 7~6 **Reserved**: Reserved bits

Bit 5~0 **RC32KCalValue**: LPCD 32kHz RC calibration value

• LPCDADCRef Register

Address	Bit	7	6	5	4	3	2	1	0
36h	Name	LPCDADCRef							
	Type	R/W							
	Reset Value	1	0	0	0	0	0	0	0

Bit 7~0 **LPCDADCRef**: LPCD ADC reference value

• CWGsN_LPCD Register

Address	Bit	7	6	5	4	3	2	1	0
38h	Name	CWGsN_LPCD[3:0]				Reserved			
	Type	R/W				—			
	Reset Value	0	0	1	1	0	0	0	0

Bit 7~4 **CWGsN_LPCD[3:0]**: Define the conductance value of N driver during the LPCD mode

Bit 3~0 **Reserved**: Reserved bits

• CWGsP_LPCD Register

Address	Bit	7	6	5	4	3	2	1	0
39h	Name	Reserved		CWGsP_LPCD[5:0]					
	Type	—		R/W					
	Reset Value	0	0	0	1	1	1	1	1

Bit 7~6 **Reserved**: Reserved bits

Bit 5~0 **CWGsP_LPCD[5:0]**: Define the conductance value of P driver during the LPCD mode

Command Set

The NFC reader operation is determined by an internal state machine capable of executing a set of commands. A command is executed by writing its command code into the Command register.

The parameters and/or data required for a command are processed by the FIFO buffer.

General Features

Each command that requires a data bit stream (or data byte stream) as input immediately processes any data in the FIFO buffer. An exception to this is the Transceive command, which starts the transmitter by setting the StartSend bit in the BitFraming register.

Each command that requires some preset parameters only starts running when the correct number of parameters are received from the FIFO buffer.

The FIFO buffer is not immediately cleared when the commands start. Therefore, it is possible to write command parameters and data to the FIFO buffer before starting the command.

Each command can be interrupted by a new command written to the Command register, such as the Idle command.

Command Overview

Command	Command Code	Action
Idle	0000	No action; cancels the current command execution
CalcCRC	0011	Activates the CRC coprocessor or performs a self-test
Transmit	0100	Transmits data from the FIFO buffer
NoCmdChange	0111	No command change; used to modify some bits of the Command register without affecting the current command, such as the PowerDown bit
Receive	1000	Activates the receiver circuit
Transceive	1100	Transmits data from the FIFO buffer to antenna and automatically activates the receiver after transmission
MFAuthent	1110	Performs the MIFARE standard authentication as a reader
SoftReset	1111	Reset the chip

Command Description

Idle Command

This command places the chip in the idle mode. The command terminates itself.

CalcCRC Command

The FIFO buffer content is transferred to the CRC coprocessor and the CRC calculation starts. The calculation result is stored in the CRCResult_H and CRCResult_L registers. The CRC calculation is not limited to some specific bytes. The calculation is not stopped when the FIFO buffer is empty during the data stream. The next byte written to the FIFO buffer is also added to the calculation.

The CRC preset value is controlled by the CRCPreset[1:0] bit field in the Mode register. The preset value is loaded into the CRC coprocessor when the command starts.

This command must be terminated by writing any command to the Command register, such as the Idle command.

Transmit Command

The FIFO buffer content is transmitted immediately after starting this command. Before transmitting the FIFO buffer content, all relevant registers must be properly configured for data transmission.

This command automatically terminates when the FIFO buffer is empty. It can be terminated by another command written to the Command register.

NoCmdChange Command

This command does not affect any running command in the Command registesr. It can be used to modify any bit in the Command register except the Command[3:0], such as RcvOff bit or PowerDown bit.

Receive Command

The receiver circuit is activated by this command and then waits for a data stream to be received. All relevant registers must be correctly configured before starting this command.

This command automatically terminates when data stream ends. This is indicated by the frame end pattern or by the length byte according to the selected frame type and speed.

Note that the Receive command will not automatically terminate if the RxMultiple bit in the RxMode register is set to 1. In this case, the command must be terminated by starting another command in the Command register.

Transceive Command

This command continuously repeats the transmission of data from the FIFO buffer and the reception of data from the RF field. The first action is transmitting and after transmission the command changes to receive a data stream.

Each transmission process is started by setting the StartSend bit in the BitFraming register to logic 1. This command must be cleared by writing any other command to the Command register.

Note that if the RxMultiple bit in the RxMode register is set to 1, the Transceive command never leaves the receiving state because this state cannot be automatically cancelled.

MFAuthent Command

This command is used to perform the Crypto_M authentication to enable secure communication with any Crypto_M common cards. The following data must be written into the FIFO buffer before starting the command.

- Authentication command code (0x60, 0x61)
- Block address
- Sector key byte 0
- Sector key byte 1
- Sector key byte 2
- Sector key byte 3
- Sector key byte 4
- Sector key byte 5
- Card serial number byte 0
- Card serial number byte 1
- Card serial number byte 2
- Card serial number byte 3

12 bytes in total must be written into the FIFO buffer.

Note that all FIFO accesses are blocked when the MFAuthent command is active. If there is an access to the FIFO buffer, the WrErr bit in the Error register will be set high.

This command automatically terminates when the Crypto_M card is authenticated and the MFCrypto1On bit in the Status2 register is set high.

This command dose not automatically terminate if the card does not respond. Therefore, the timer must be initialised to the automatic mode. In this case, in addition to the IdleIRq flag, the TimerIRq flag can also be used as the command termination criterion. During the authentication process, the RxIRq and TxIRq flags are blocked. The MFCrypto1On bit is valid only after the authentication command is completed, either after processing the agreement or after writing the Idle command code to the Command register.

If an error occurs during the authentication process, the ProtocolErr bit in the Error is set high and the MFCrypto1On bit in the Status2 register is cleared to zero.

SoftReset Command

This command performs a reset to the chip. The data in the internal buffer remains unchanged and all registers are set to the reset values. The command automatically terminates when finished.

Note: The SerialSpeed register is reset and therefore the serial data rate is set to 9.6kbit/s.

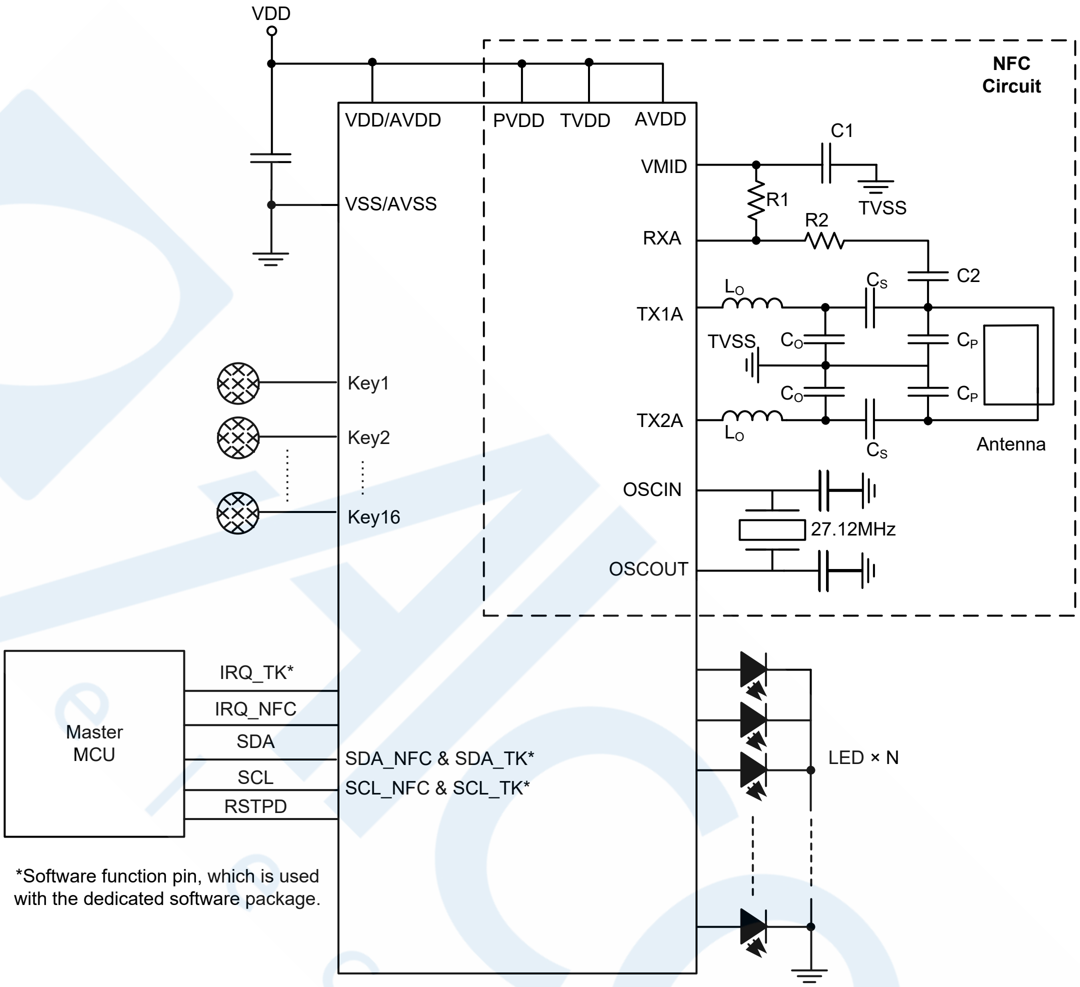
Configuration Options

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. The option must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Option	
1	HIRC frequency selection – f_{HIRC}: 8MHz, 12MHz, 16MHz

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

Application Circuits



*Software function pin, which is used with the dedicated software package.

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

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

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

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

Extended Instruction Set

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

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

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

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

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

Instruction Definition

ADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C, SC

ADCM A,[m]	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C, SC

ADD A,[m]	Add Data Memory to ACC
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m]
Affected flag(s)	OV, Z, AC, C, SC

ADD A,x	Add immediate data to ACC
Description	The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + x
Affected flag(s)	OV, Z, AC, C, SC

ADDM A,[m]	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m]
Affected flag(s)	OV, Z, AC, C, SC

AND A,[m]	Logical AND Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC “AND” [m]
Affected flag(s)	Z

AND A,x	Logical AND immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC “AND” x
Affected flag(s)	Z

ANDM A,[m]	Logical AND ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.
Operation	[m] ← ACC “AND” [m]
Affected flag(s)	Z

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None

CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None

CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None

CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	[m] ← [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 ← [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] − 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] – 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] − C
Affected flag(s)	OV, Z, AC, C, SC, CZ

SDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] − 1 Skip if [m]=0
Affected flag(s)	None

SDZA [m]	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	ACC ← [m] − 1 Skip if ACC=0
Affected flag(s)	None

SET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	[m] ← FFH
Affected flag(s)	None

SET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	[m].i ← 1
Affected flag(s)	None

SIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] + 1 Skip if [m]=0
Affected flag(s)	None

SIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] + 1 Skip if ACC=0
Affected flag(s)	None

SNZ [m].i	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None

SNZ [m]	Skip if Data Memory is not 0
Description	The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m]≠ 0
Affected flag(s)	None

SUB A,[m]	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m]
Affected flag(s)	OV, Z, AC, C, SC, CZ

SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m]
Affected flag(s)	OV, Z, AC, C, SC, CZ

SUB A,x	Subtract immediate data from ACC
Description	The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − x
Affected flag(s)	OV, Z, AC, C, SC, CZ

SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0 ↔ [m].7~[m].4
Affected flag(s)	None

SWAPA [m]	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3~ACC.0 ← [m].7~[m].4 ACC.7~ACC.4 ← [m].3~[m].0
Affected flag(s)	None

SZ [m]	Skip if Data Memory is 0
Description	The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m]=0
Affected flag(s)	None

SZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] Skip if [m]=0
Affected flag(s)	None

SZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if [m].i=0
Affected flag(s)	None

TABRD [m]	Read table (specific page) to TBLH and Data Memory
Description	The low byte of the program code (specific page) addressed by the table pointer (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

TABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

ITABRD [m]	Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

ITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "XOR" [m]
Affected flag(s)	Z

XORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC "XOR" [m]
Affected flag(s)	Z

XOR A,x	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "XOR" x
Affected flag(s)	Z

Extended Instruction Definition

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

LADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C, SC

LADCM A,[m]	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C, SC

LADD A,[m]	Add Data Memory to ACC
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m]
Affected flag(s)	OV, Z, AC, C, SC

LADDM A,[m]	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m]
Affected flag(s)	OV, Z, AC, C, SC

LAND A,[m]	Logical AND Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC “AND” [m]
Affected flag(s)	Z

LANDM A,[m]	Logical AND ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.
Operation	[m] ← ACC “AND” [m]
Affected flag(s)	Z

LCLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None

LCLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None

LCPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	[m] ← [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 ← [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] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← ACC + 66H
Affected flag(s)	C

LDEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [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 ← [m] − 1
Affected flag(s)	Z

LINC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [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 ← [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 theAccumulator 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] − 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] − C
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] − 1 Skip if [m]=0
Affected flag(s)	None

LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	ACC ← [m] − 1 Skip if ACC=0
Affected flag(s)	None

LSET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	[m] ← FFH
Affected flag(s)	None

LSET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	[m].i ← 1
Affected flag(s)	None

LSIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] + 1 Skip if [m]=0
Affected flag(s)	None

LSIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] + 1 Skip if ACC=0
Affected flag(s)	None

LSNZ [m].i	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None

LSNZ [m]	Skip if Data Memory is not 0
Description	The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the content of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m] ≠ 0
Affected flag(s)	None

LSUB A,[m]	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m]
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m]
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0 ↔ [m].7~[m].4
Affected flag(s)	None

LSWAPA [m]	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3~ACC.0 ← [m].7~[m].4 ACC.7~ACC.4 ← [m].3~[m].0
Affected flag(s)	None

LSZ [m]	Skip if Data Memory is 0
Description	The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m]=0
Affected flag(s)	None

LSZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] Skip if [m]=0
Affected flag(s)	None

LSZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if [m].i=0
Affected flag(s)	None

LTABRD [m]	Read table (specific page) to TBLH and Data Memory
Description	The low byte of the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

LTABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

LITABRD [m]	Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

LITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

LXOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC “XOR” [m]
Affected flag(s)	Z

LXORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC “XOR” [m]
Affected flag(s)	Z

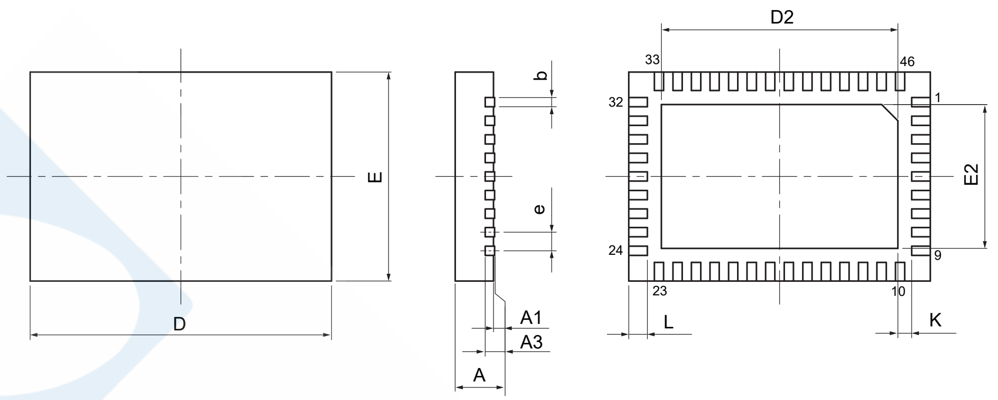
Package Information

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

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

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

SAW Type 46-pin QFN (6.5mm×4.5mm×0.75mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3		0.008 REF	
b	0.006	0.008	0.010
D		0.256 BSC	
E		0.177 BSC	
e		0.016 BSC	
D2	0.197	—	0.205
E2	0.118	—	0.126
L	0.014	0.016	0.018
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3		0.203 REF	
b	0.15	0.20	0.25
D		6.50 BSC	
E		4.50 BSC	
e		0.40 BSC	
D2	5.00	—	5.20
E2	3.00	—	3.20
L	0.35	0.40	0.45
K	0.20	—	—

Singel 3 | B-2550 Kontich | Belgium | Tel. +32 (0)3 458 30 33
info@alcom.be | www.alcom.be
Rivium 1e straat 52 | 2909 LE Capelle aan den IJssel | The Netherlands
Tel. +31 (0)10 288 25 00 | info@alcom.nl | www.alcom.nl

Copyright© 2024 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

The information provided in this document has been produced with reasonable care and attention before publication, however, HOLTEK does not guarantee that the information is completely accurate. The information contained in this publication is provided for reference only and may be superseded by updates. HOLTEK disclaims any expressed, implied or statutory warranties, including but not limited to suitability for commercialization, satisfactory quality, specifications, characteristics, functions, fitness for a particular purpose, and non-infringement of any third-party's rights. HOLTEK disclaims all liability arising from the information and its application. In addition, HOLTEK does not recommend the use of HOLTEK's products where there is a risk of personal hazard due to malfunction or other reasons. HOLTEK hereby declares that it does not authorise the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.