Smart Card Reader Flash MCU

HT66F4360/HT66F4370/HT66F4390

Revision: V1.80 Date: November 19, 2019

www.holtek.com

Table of Contents

Features

CPU Features

- Operating voltage
  - V_{DD} (MCU)
    - f_{SYS}= 6MHz: 2.2V~5.5V
    - f_{SYS}= 12MHz: 2.7V~5.5V
  - V_{DD} (USB Mode)
    - f_{SYS}= 6MHz/12MHz: 2.7V~5.5V
    - f_{SYS}= 16MHz: 4.5V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator Types
  - External High Speed Crystal – HXT
  - Internal High Speed RC – HIRC
  - External 32.768kHz Crystal – LXT
  - Internal 32kHz RC – LIRC
- Fully integrated internal 12 MHz oscillator with 0.25% accuracy for all USB modes which requires no external components
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one to three instruction cycles
- Table read instructions
- 115 powerful instructions
- Up to 16-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Program Memory: Up to 64K×16
- Data Memory: 3072×8
- True EEPROM Memory: 256×8 (Only for HT66F4390)
- Watchdog Timer function
- Up to 36 bidirectional I/O lines
- Dual external interrupt lines shared with I/O pins
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
  - Dual Compact type 10-bit Timer Modules – CTM0 & CTM1
  - Single Periodic type 10-bit Timer Module – PTM
  - Single Standard type 16-bit Timer Module – STM
- Smart Card interface compatible with and certifiable to the ISO 7816-3 Standards
- 1.8V, 3V, 5V smart card supply
- USB interface
  - USB 2.0 Full Speed compatible
  - 8 endpoints supported including endpoint 0
  - All endpoints except endpoint 0 can support interrupt and bulk transfer
  - All emdpoints except endpoint 0 can be configured as 8, 16, 32, 64 bytes FIFO size
  - Endpoint 0 supports control transfer
  - Endpoint 0 FIFO size: 8 bytes
  - Integrated an internal 1.5kΩ pull-high resistor on UDP pin
  - Support 3.3V LDO
- Dual Serial SPI Interfaces – SPI0 & SPI1
- Single I^2C interface
- Dual Fully-duplex Universal Asynchronous Receiver and Transmitter Interface – UART0 & UART1
- Dual comparator functions
- Dual Time-Base functions for generation of fixed time interrupt signals
- 8-channel 12-bit resolution A/D converter
- In Application Programming function – IAP
- In System Programming function – ISP
- Low voltage reset function
- Low voltage detect function
- Flash program memory can be re-programmed up to 10,000 times
- Flash program memory data retention > 10 years
- EEPROM data memory can be re-programmed up to 100,000 times (Only for HT66F4390)
- EEPROM data memory data retention > 10 years (Only for HT66F4390)
- Package types: 48-pin LQFP, 64-pin LQFP

General Description

The devices are Flash Memory A/D type 8-bit high performance RISC architecture microcontrollers with fully integrated Smart Card interface function, designed for applications that communicate with various types of Smart Card. Offering users the convenience of Flash Memory multi-programming features, these devices also include a wide range of functions and features. In addition to the flash program memory, other memory includes an area of RAM Data Memory. The HT66F4390 also inculdes true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 12-bit A/D converter, dual comparator functions, a DC/DC converter and an LDO. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Easy communication with the outside world is provided using the internal fully integrated SPI, I^2C, UART and USB interface functions, four popular interfaces which provide designers with a means of easy communication with external peripheral hardware. The inclusion of flexible I/O programming features, Time Base functions together with many other features further enhance device functionality and flexibility for wide range of application possibilities. 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.

These devices also include fully integrated low and high speed oscillators which are flexibly used for different applications. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

The unique Holtek TinyPower technology also offers the advantages of extremely low current consumption characteristics, an extremely important consideration in the present trend for low power battery powered applications. The usual Holtek MCU features such as power down and wake-up functions, oscillator options, programmable frequency divider, etc., combine to ensure user applications require a minimum of external components.

Selection Table

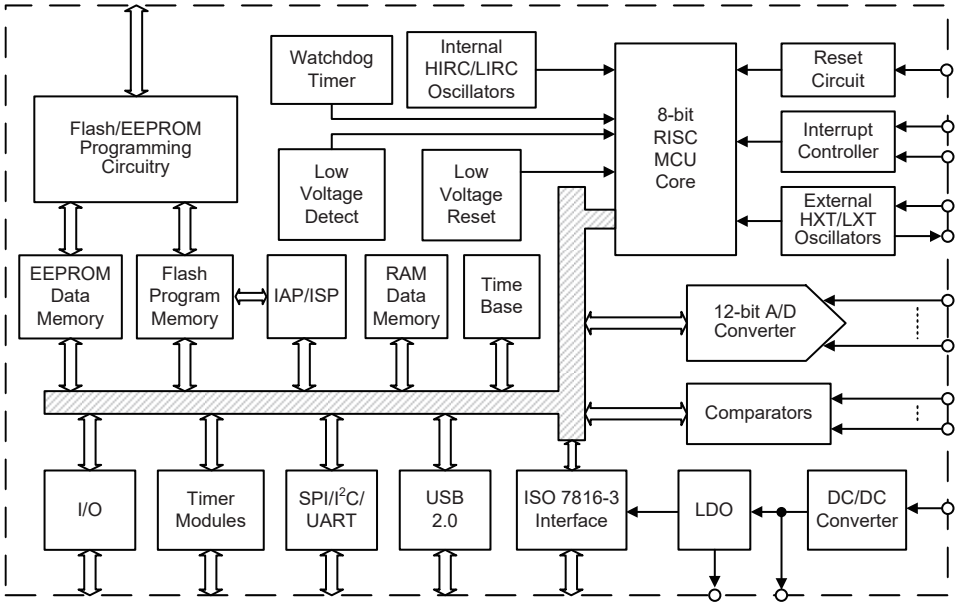
Most features are common to all devices. The main features distinguishing them are the Program Memory capacity, EEPROM Memory and stack capacity. The following table summarises the main features of each device.

Part No.	Program Memory	Data Memory	EEPROM	I/O	External Interrupt	A/D	Timer Module
HT66F4360	16K×16	3072×8	—	36	2	12-bit×8	10-bit CTM×2 10-bit PTM×1 16-bit STM×1
HT66F4370	32K×16		—				
HT66F4390	64K×16		256×8				

Part No.	Time Base	SPI	I^2C	UART	USB	Smart Card	Stacks	Package
HT66F4360	2	2	√	√	√	√	12	48/64LQFP
HT66F4370							12	
HT66F4390							16	

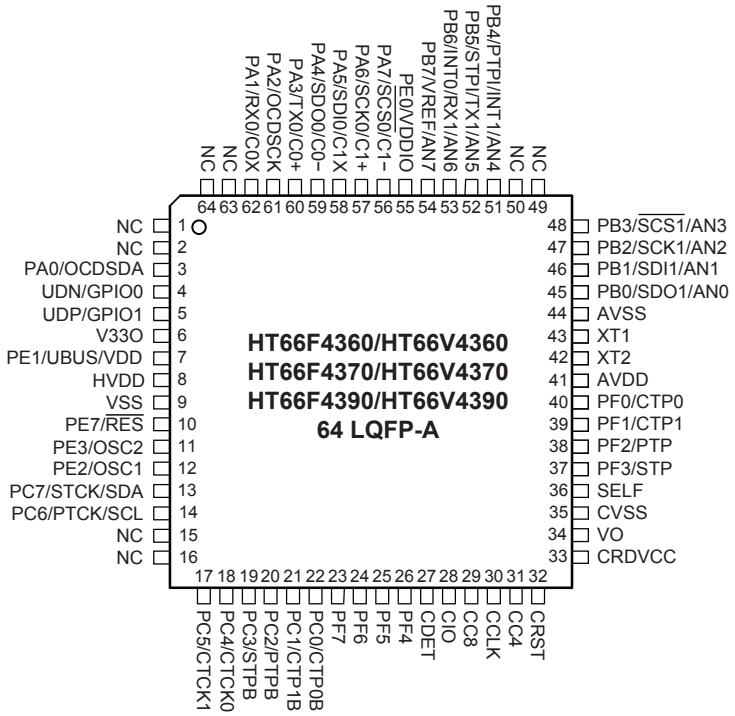
Note: As devices exist in more than one package format, the table reflects the situation for the package with the most pins.

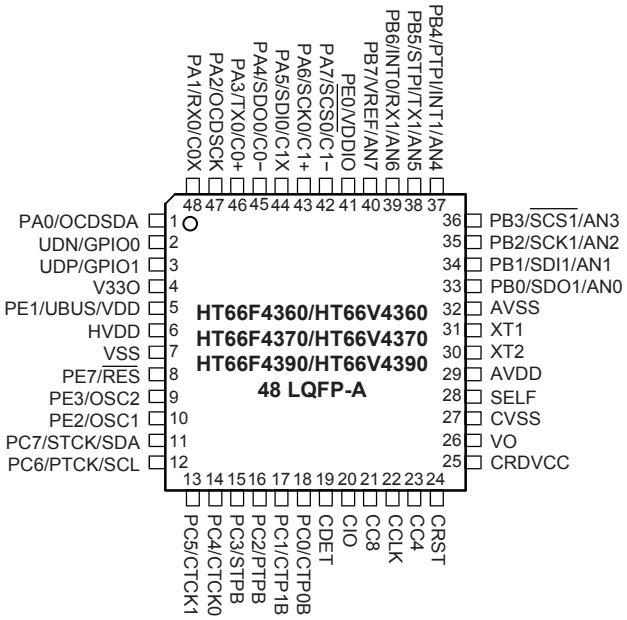
Block Diagram



Note: The EEPROM Data Memory is only for HT66F4390.

Pin Assignment





Note: 1. If the pin-shared pin functions have multiple outputs simultaneously, its pin names at the right side of the "/" sign can be used for higher priority.

2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the HT66V4360/4370/4390 device which is the OCDS EV chip for the HT66F4360/4370/4390 device.

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

Pin Descriptions

With the exception of the power pins and some relevant transformer control pins, all pins on these devices can be referenced by their Port name, e.g. PA.0, PA.1 etc, which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Analog to Digital Converter, Timer Module pins etc. The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pad Name	Function	OPT	I/T	O/T	Description
PA0/OCDSDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OCDSDA	—	ST	CMOS	OCDS Data/Address pin, for EV chip only.
PA1/RX0/C0X	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	RX0	PAS0	ST	—	UART0 RX serial data input
	C0X	PAS0	—	CMOS	Comparator 0 output
PA2/OCDSCK	PA2	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OCDSCK	—	ST	CMOS	OCDS Clock pin, for EV chip only.

Pad Name	Function	OPT	I/T	O/T	Description
PA3/TX0/C0+	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TX0	PAS0	—	CMOS	UART0 TX serial data output
	C0+	PAS0	AN	—	Comparator 0 positive input
PA4/SDO0/C0−	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDO0	PAS1	—	CMOS	SPI0 data output
	C0−	PAS1	AN	—	Comparator 0 negative input
PA5/SDI0/C1X	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDI0	PAS1	ST	—	SPI0 data input
	C1X	PAS1	—	CMOS	Comparator 1 output
PA6/SCK0/C1+	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCK0	PAS1	ST	CMOS	SPI0 serial clock
	C1+	PAS1	AN	—	Comparator 1 positive input
PA7/$\overline{\text{SCS0}}$/C1−	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{\text{SCS0}}$	PAS1	ST	CMOS	SPI0 slave select
	C1−	PAS1	AN	—	Comparator 1 negative input
PB0/SDO1/AN0	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SDO1	PBS0	—	CMOS	SPI1 data output
	AN0	PBS0	AN	—	A/D Converter analog input
PB1/SDI1/AN1	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SDI1	PBS0	ST	—	SPI1 data input
	AN1	PBS0	AN	—	A/D Converter analog input
PB2/SCK1/AN2	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SCK1	PBS0	ST	CMOS	SPI1 serial clock
	AN2	PBS0	AN	—	A/D Converter analog input
PB3/$\overline{\text{SCS1}}$/AN3	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{\text{SCS1}}$	PBS0	ST	CMOS	SPI1 slave select
	AN3	PBS0	AN	—	A/D Converter analog input
PB4/PTPI/INT1/AN4	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTPI	PBS1	ST	—	PTM capture input
	INT1	PBS1 INTEG INTC1	ST	—	External Interrupt 1
	AN4	PBS1	AN	—	A/D Converter analog input

Pad Name	Function	OPT	I/T	O/T	Description
PB5/STPI/TX1/AN5	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	STPI	PBS1	ST	—	STM capture input
	TX1	PBS1	—	CMOS	UART1 TX serial data output
	AN5	PBS1	AN	—	A/D Converter analog input
PB6/INT0/RX1/AN6	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	INT0	PBS1 INTEG INTC0	ST	—	External Interrupt 0
	RX1	PBS1	ST	—	UART1 RX serial data input
	AN6	PBS1	AN	—	A/D Converter analog input
PB7/VREF/AN7	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	VREF	PBS1	AN	—	A/D Converter reference voltage input
	AN7	PBS1	AN	—	A/D Converter analog input
PC0/CTP0B	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0B	PCS1	—	CMOS	CTM0 inverted output
PC1/CTP1B	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP1B	PCS1	—	CMOS	CTM1 inverted output
PC2/PTPB	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTPB	PCS0	—	CMOS	PTM inverted output
PC3/STPB	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	STPB	PCS0	—	CMOS	STM inverted output
PC4/CTCK0	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTCK0	PCS1	ST	—	CTM0 clock input
PC5/CTCK1	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTCK1	PCS1	ST	—	CTM1 clock input
PC6/PTCK/SCL	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTCK	PCS1	ST	—	PTM clock input
	SCL	PCS1	ST	NMOS	I^2C clock line
PC7/STCK/SDA	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	STCK	PCS1	ST	—	STM clock input
	SDA	PCS1	ST	NMOS	I^2C data/address line
PE0/VDDIO	PE0	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	VDDIO	PES0	PWR	—	PA & PB external power supply
PE1/UBUS/VDD	PE1	—	ST	—	General purpose Input
	UBUS	—	PWR	—	USB SIE power supply
	VDD	—	PWR	—	Power supply
PE2/OSC1	PE2	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OSC1	PES0	HXT	—	HXT oscillator pin

Pad Name	Function	OPT	I/T	O/T	Description
PE3/OSC2	PE3	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	OSC2	PES0	—	HXT	HXT oscillator pin
PE7/$\overline{\text{RES}}$	PE7	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	$\overline{\text{RES}}$	RSTC	ST	—	External reset pin
PF0/CTP0	PF0	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0	PFS0	—	CMOS	CTM0 output
PF1/CTP1	PF1	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP1	PFS0	—	CMOS	CTM1 output
PF2/PTP	PF2	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP	PFS0	—	CMOS	PTM output
PF3/STP	PF3	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	STP	PFS0	—	CMOS	STM output
PF4~PF7	PFn	—	ST	CMOS	General purpose I/O. Register enabled pull-up.
XT1	XT1	—	LXT	—	LXT oscillator pin
XT2	XT2	—	—	LXT	LXT oscillator pin
UDN/GPIO0	UDN	—	ST	CMOS	USB UDN line
	GPIO0	—	ST	CMOS	General purpose I/O
UDP/GPIO1	UDP	—	ST	CMOS	USB UDP line
	GPIO1	—	ST	CMOS	General purpose I/O
V33O	V33O	—	—	PWR	3.3V regulator output
CRST	CRST	—	—	CMOS	Smart Card reset output
CCLK	CCLK	—	—	CMOS	Smart Card clock output
CIO	CIO	—	ST	CMOS	Smart Card data input/output
CDET	CDET	—	ST	—	Smart Card detection input
CC4	CC4	—	ST	CMOS	Smart Card C4 input/output
CC8	CC8	—	ST	CMOS	Smart Card C8 input/output
CRDVCC	CRDVCC	—	PWR	—	Positive power supply for external smart cards
SELF	SELF	—	PWR	—	External inductor connection pin for DC/DC converter
VO	VO	—	PWR	—	External diode connection pin for DC/DC converter
CVSS	CVSS	—	PWR	—	DC/DC converter negative power supply, ground.
VSS	VSS	—	PWR	—	Negative power supply, ground.
AVDD	AVDD	—	PWR	—	Analog positive power supply
AVSS	AVSS	—	PWR	—	Analog negative power supply, ground.
HVDD	HVDD	—	PWR	—	HIRC oscillator positive power supply
NC	NC	—	—	—	Non-connected.

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

Absolute Maximum Ratings

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

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (HXT)	—	f_{SYS}=6MHz	2.2	—	5.5	V
			f_{SYS}=12MHz	2.7	—	5.5	V
			f_{SYS}=16MHz	4.5	—	5.5	V
	Operating Voltage (HIRC)	—	f_{SYS}=12MHz	2.7	—	5.5	V
	Operating Voltage (LXT)	—	f_{SYS}=f_{LXT}=32.768kHz	2.2	—	5.5	V
	Operating Voltage (LIRC)	—	f_{SYS}=f_{LIRC}=32kHz	2.2	—	5.5	V
I_{DD}	Operating Current (HXT)	3V	f_{SYS}=f_{HXT}=6MHz No load, all peripherals off	—	0.7	1.5	mA
		5V		—	1.6	3.0	mA
		3V	f_{SYS}=f_{HXT}=12MHz No load, all peripherals off	—	1.3	3.0	mA
		5V		—	2.7	6.0	mA
		5V	f_{SYS}=f_{HXT}=16MHz No load, all peripherals off	—	3.6	8.0	mA
	Operating Current (HIRC)	3V	f_{SYS}=f_{HIRC}=12MHz No load, all peripherals off	—	1.6	3.5	mA
		5V		—	2.8	6.0	mA
	Operating Current (LXT)	3V	f_{SYS}=f_{SUB}=f_{LXT}=32.768kHz No load, all peripherals off	—	17	30	μA
		5V		—	30	50	μA
	Operating Current (LIRC)	3V	f_{SYS}=f_{SUB}=f_{LIRC}=32kHz No load, all peripherals off	—	16	30	μA
		5V		—	28	50	μA
	Operating Current (USB)	3V	f_{SYS}=f_{PLL}/8=6MHz, No load, USB and PLL on, other peripherals off	—	4.5	9	mA
		5V		—	9.5	15	mA
		3V	f_{SYS}=f_{PLL}/4=12MHz, No load, USB and PLL on, other peripherals off	—	5	10	mA
		5V		—	10.5	16.0	mA
		5V	f_{SYS}=f_{PLL}/3=16MHz, No load, USB and PLL on, other peripherals off	—	12	18	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB}	Standby Current (IDLE0 Mode)	3V	f_{SYS} off, $f_{SUB}=f_{LXT}$ on, No load, all peripherals off	—	1.2	2.0	μA
		5V		—	1.5	3.0	μA
		3V	f_{SYS} off, $f_{SUB}=f_{LIRC}$ on No load, all peripherals off	—	1.5	3.0	μA
		5V		—	2.8	5.0	μA
	Standby Current (IDLE1 Mode)	3V	$f_{SYS}= f_{HXT}$=12MHz on, f_{SUB} on, No load, all peripherals off	—	0.65	1.40	mA
		5V		—	1.3	3.0	mA
		3V	$f_{SYS}= f_{HIRC}$=12MHz on, f_{SUB} on, No load, all peripherals off	—	0.9	1.5	mA
		5V		—	1.4	3.0	mA
	Standby Current (SLEEP Mode)	3V	f_{SYS} off, f_{SUB} off, No load, all peripherals off, WDT disabled	—	0.14	0.19	μA
		5V		—	0.21	0.50	μA
		3V	f_{SYS} off, $f_{SUB}=f_{LXT}$ on No load, all peripherals off, WDT enabled	—	1.0	1.5	μA
		5V		—	1.3	3.0	μA
		3V	f_{SYS} off, $f_{SUB}=f_{LIRC}$ on, No load, all peripherals off, WDT enabled	—	1.2	3.0	μA
		5V		—	2.1	5.0	μA
I_{SUS}	Suspend Current (USB)	5V	f_H=off, SUSP2=0, RCTRL=0, V33O on, MCU powered down, No load, USB and PLL on, other peripherals off	—	330	450	μA
		5V	f_H=off, SUSP2=1, RCTRL=1, V33O off, MCU powered down, No load, USB and PLL on, other peripherals off	—	240	330	μA
V_{IL}	Input Low Voltage for I/O Ports or Input Pins	5V	—	0.0	—	1.5	V
		—	—	0.0	—	$0.2V_{DD}$	V
	Input Low Voltage for $\overline{RES}$ Pin	—	—	0.0	—	$0.4V_{DD}$	V
V_{IH}	Input High Voltage for I/O Ports or Input Pins	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	V
	Input High Voltage for $\overline{RES}$ Pin	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	Sink Current for I/O Pins	3V	$V_{OL}=0.1V_{DD}$	17	34	—	mA
		5V		34	68	—	mA
I_{OH}	Source Current for I/O Pins	3V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	mA
V_{OH}	Output High Voltage for I/O Ports	3V	I_{OH}= mA	2.7	—	—	V
		5V	I_{OH}=5mA	4.5	—	—	V
V_{OL}	Output Low Voltage for I/O Ports	3V	I_{OL}=4mA	—	—	0.3	V
		5V	I_{OL}=10mA	—	—	0.5	V
R_{PH}	Pull-high Resistance for I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
I_{LEAK}	Input Leakage Current	3V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
		5V		—	—	±1	μA
V_{33O}	3.3V Regulator Output Voltage	5V	I_{V33O}=70mA	3.0	3.3	3.6	V
R_{UDP1}	Pull-high Resistance between UDP and V33O	3.3V	—	Typ. -5%	1.5	Typ. +5%	kΩ
R_{UDP2}	Pull-high Resistance between UDP and UBUS	5V	—	Typ. -40%	7.5	Typ. +40%	kΩ
R_{UDPN}	Pull-high Resistance between UDP/UDN and UBUS	5V	—	300	600	800	kΩ
R_{PL}	Pull-low Resistance to UBUS	5V	SUSP2=1, RUBUS=0	0.5	1.0	1.5	MΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
f_{SYS}	System Clock (HXT)	2.2V~5.5V	$f_{SYS}=f_{HXT}$=6MHz	—	6	—	MHz
		2.7V~5.5V	$f_{SYS}=f_{HXT}$=12MHz	—	12	—	MHz
		4.5V~5.5V	$f_{SYS}=f_{HXT}$=16MHz	—	16	—	MHz
	System Clock (HIRC)	2.7V~5.5V	$f_{SYS}=f_{HIRC}$=12MHz	—	12	—	MHz
	System Clock (LXT)	2.2V~5.5V	$f_{SYS}=f_{LXT}$=32.768kHz	—	32.768	—	kHz
	System Clock (LIRC)	2.2V~5.5V	$f_{SYS}=f_{LIRC}$=32kHz	—	32	—	kHz
f_{HIRC}	High Speed Internal RC Oscillator (HIRC)	2.7V~5.5V	Non-USB mode, Ta=25°C	Typ. -3%	12	Typ. +3%	MHz
		3.0V~5.5V	Non-USB mode, Ta=0°C~70°C	Typ. -6%	12	Typ. +6%	MHz
		2.7V~5.5V	Non-USB mode, Ta=-40°C~85°C	Typ. -10%	12	Typ. +10%	MHz
		3.3V~5.5V	USB mode, Ta=25°C	Typ. -0.25%	12	Typ. +0.25%	MHz
f_{LIRC}	Low Speed Internal RC Oscillator (LIRC)	5V	Ta=25°C	Typ. -10%	32	Typ. +10%	kHz
		5V±0.5V	Ta= -40°C to 85°C	Typ. -40%	32	Typ. +40%	kHz
		2.2V~5.5V	Ta= -40°C to 85°C	Typ. -50%	32	Typ. +60%	kHz
t_{TCK}	CTCKn, STCK, PTCK pin Minimum Input Pulse Width	—	—	0.3	—	—	μs
t_{TPI}	STPI, PTPI pin Minimum Input Pulse Width	—	—	0.3	—	—	μs
t_{INT}	Interrupt Pin Minimum Input Pulse Width	—	—	10	—	—	μs
t_{RES}	External Reset Pin Minimum Input Low Pulse Width	—	—	10	—	—	μs
t_{SST}	System Start-up Timer Period (Wake-up from Power Down Mode and f_{SYS} off or $\overline{RES}$ Pin Reset)	—	$f_{SYS}=f_H=f_{HXT}$~f_{HXT}/64	128	—	—	t_{HXT}
		—	$f_{SYS}=f_H=f_{HIRC}$~f_{HIRC}/64	16	—	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LXT}$	1024	—	—	t_{LXT}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	2	—	—	t_{LIRC}
	System Start-up Timer Period (Wake-up from Power Down Mode and f_{SYS} on)	—	$f_{SYS}=f_H$~f_H/64, $f_H=f_{HXT}$ or f_{HIRC}	2	—	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LXT}$ or f_{LIRC}	2	—	—	t_{SUB}
	System Start-up Timer Period (SLOW Mode→NORMAL Mode) (NORMAL Mode→SLOW Mode) (f_{HXT}/f_{HIRC} Switch, f_{LXT}/f_{LIRC} Switch)	—	f_{HXT} off→on (HXTF=1)	1024	—	—	t_{HXT}
		—	f_{HIRC} off→on (HIRCF=1)	16	—	—	t_{HIRC}
		—	f_{LXT} off→on (LXTF=1)	1024	—	—	t_{LXT}
	System Start-up Timer Period (WDT Hardware Reset)	—	—	0	—	—	t_{SYS}
t_{RSTD}	System Reset Delay Time (Power-on Reset, LVR Hardware Reset, LVRC/WDTC/RSTC Software Reset)	—	—	25	50	100	ms
	System Reset Delay Time ($\overline{RES}$ pin Reset, USB Reset WDT Hardware Reset)	—	—	8.3	16.7	33.3	ms
t_{DEW}	EEPROM write time (HT66F4390)	—	—	—	4	6	ms

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

LVD/LVR Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR Enable, voltage select 2.1V	Typ. -5%	2.1	Typ. +5%	V
			LVR Enable, voltage select 2.55V		2.55		
			LVR Enable, voltage select 3.15V		3.15		
			LVR Enable, voltage select 3.8V		3.8		
V_{LVD}	Low Voltage Detector Voltage	—	LVD Enable, voltage select 2.0V	Typ. -5%	2.0	Typ. +5%	V
			LVD Enable, voltage select 2.2V		2.2		
			LVD Enable, voltage select 2.4V		2.4		
			LVD Enable, voltage select 2.7V		2.7		
			LVD Enable, voltage select 3.0V		3.0		
			LVD Enable, voltage select 3.3V		3.3		
			LVD Enable, voltage select 3.6V		3.6		
			LVD Enable, voltage select 4.0V		4.0		
V_{BG}	Bandgap Reference Voltage	—	—	Typ. -5%	1.04	Typ. +5%	V
$I_{LVRLVDBG}$	LVD/LVR Operating Current	3V	LVD/LVR Enable, VBGEN=0	—	15	20	μA
		5V	LVD/LVR Enable, VBGEN=0	—	20	30	μA
		3V	LVD/LVR Enable, VBGEN=1	—	100	150	μA
		5V	LVD/LVR Enable, VBGEN=1	—	145	200	μA
t_{BGS}	V_{BG} Turn on Stable Time	—	No load	—	—	150	μs
t_{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off→on	—	—	15	μs
		—	For LVR disable, VBGEN=0, LVD off→on	—	—	150	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs
I_{LVR}	Additional Current for LVR Enabled	5V	LVD disable, VBGEN=0	—	14	20	μA
I_{LVD}	Additional Current for LVD Enabled	5V	LVR disable, VBGEN=0	—	7	15	μA
I_{BG}	Additional Current for Bandgap Circuit Enabled	3V	LVR disable, LVD disable	—	85	150	μA
		5V		—	125	200	μA

A/D Converter Electrical Characteristics

Operating Temperature: -40°C~85°C, unless otherwise specified.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
AV_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	2	—	AV_{DD}	V
DNL	Differential Non-linearity	2.2V~2.7V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=8.0μs	-15	—	+15	LSB
		2.7V~5.5V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=0.5μs	-3	—	+3	LSB
INL	Integral Non-linearity	2.2V~2.7V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=8.0μs	-16	—	+16	LSB
		2.7V~5.5V	$V_{REF}=AV_{DD}=V_{DD}$, t_{ADCK}=0.5μs	-4	—	+4	LSB
I_{ADC}	Additional Current Consumption for A/D Converter Enabled	3V	No load, t_{ADCK} =0.5μs	—	1.0	2.0	mA
		5V		—	1.5	3.0	mA
t_{ADCK}	Clock Period	2.2V~2.7V	—	8	—	10	μs
		2.7V~5.5V	—	0.5	—	10	μs
t_{ADC}	Conversion Time (Including A/D Sample and Hold Time)	—	—	16	—	20	t_{ADCK}
t_{ADS}	A/D Converter Sampling Time	—	—	—	4	—	t_{ADCK}
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
GERR	Gain Error	3V	$V_{REF}=AV_{DD}$	-4	—	+4	LSB
		5V					
OSRR	Offset Error	3V	$V_{REF}=AV_{DD}$	-4	—	+4	LSB
		5V					

Comparator Electrical Characteristics

Ta=25°C, All measurement is under CMPnINP input voltage=(V_{DD}-1.4)/2 and remain constant

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.7	—	5.5	V
I_{CMP}	Additional Current for Comparator Enabled	3V	CNVTn[1:0]=00B	—	1.8	3.0	μA
		5V		—	1.9	3.0	μA
V_{OS}	Input Offset Voltage	3V	Without Calibration, CNVTn[1:0]=00B, CnOF[4:0]=10000B	-10	—	10	mV
		5V	Without Calibration, CNVTn[1:0]=00B, CnOF[4:0]=10000B	-10	—	10	mV
		3V	With Calibration, CNVTn[1:0]=00B	-4	—	4	mV
		5V	With Calibration, CNVTn[1:0]=00B	-4	—	4	mV
V_{CM}	Common Mode Voltage	—	CNVTn[1:0]=00B	V_{SS}	—	V_{DD}-1.4	V
V_{HYS}	Hysteresis	3V	CNVTn[1:0]=00B	10	24	30	mV
		5V		10	24	30	mV
A_{OL}	Open Loop Gain	3V	CNVTn[1:0]=00B	60	—	—	dB
		5V		60	80	—	dB
t_{RP}	Response Time	3V	With 10mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=00B	—	—	35	μs
		5V	With 10mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=00B	—	—	35	μs
		3V	With 100mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=00B	—	—	3	μs
		5V	With 100mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=00B	—	—	3	μs
		3V	With 10mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=01B	—	—	4	μs
		5V	With 10mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=01B	—	—	4	μs
		3V	With 100mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=01B	—	—	0.6	μs
		5V	With 100mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=01B	—	—	0.6	μs
		3V	With 10mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=10B	—	—	2	μs
		5V	With 10mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=10B	—	—	2	μs
		3V	With 100mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=10B	—	—	0.3	μs
		5V	With 100mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=10B	—	—	0.3	μs
		3V	With 10mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=11B	—	—	1	μs
		5V	With 10mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=11B	—	—	1	μs
		3V	With 100mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=11B	—	—	0.25	μs
		5V	With 100mV overdrive, C_{LOAD}=3pF, CNVTn[1:0]=11B	—	—	0.25	μs

DC/DC Converter & LDO Electrical Characteristics

Ta=25°C, V_{DD}=V_{IN}, L=100μH, Current rating ≥ 100mA, DC resistance ≤ 2Ω

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IN}	DC/DC Input Voltage Range (AVDD Pin)	—	3V and 5V Regulator output, DCEN=1	2.8	—	5.5	V
		—	1.8V Regulator output, DCEN=1	2.8	—	3.4	V
		—	1.8V Regulator output, DCEN=0	3.4	—	5.5	V
V_{OUT_DCDC}	DC/DC Output Voltage (VO Pin)	—	V_{IN}=2.8V~4.0V, VSEL=0, VC[1:0]=10B, I_{LOAD}=55mA	Typ. -5%	3.8	Typ. +5%	V
		—	V_{IN}=2.8V ~ 5.5V, VSEL=1, VC[1:0]=11B, I_{LOAD}=55mA	Typ. -5%	5.5	Typ. +5%	V
V_{OUT_LDO}	LDO Output Voltage (CRDVCC Pin)	—	V_{IN}=2.8V, V_{OUT_DCDC}=3.8V, VC[1:0]=01B, I_{LOAD}=35mA	1.66	1.8	1.94	V
		—	V_{IN}=2.8V, V_{OUT_DCDC}=3.8V, VC[1:0]=10B, I_{LOAD}=55mA	2.76	3.00	3.24	V
		—	V_{IN}=2.8V, V_{OUT_DCDC}=5.5V, VC[1:0]=11B, I_{LOAD}=55mA	4.6	5.0	5.4	V
I_{OUT}	LDO Output Current	—	VC[1:0]=01B, ΔV_{OUT_LDO}=-3%	35	—	—	mA
		—	VC[1:0]=10B, ΔV_{OUT_LDO}=-3%	55	—	—	mA
		—	VC[1:0]=11B, ΔV_{OUT_LDO}=-3%	55	—	—	mA
I_Q	LDO Quiescent Current	—	VC[1:0]=11B, no load	—	190	250	μA
$\Delta V_{OUT\_LDO\_RIPPLE}$	LDO Output Voltage Ripple (CRDVCC Pin)	—	V_{IN} =2.8V, V_{OUT_DCDC}=3.8V, DCEN=1, VC[1:0]=01B, I_{LOAD}=35mA	—	—	200	mV
		—	V_{IN} =2.8V, V_{OUT_DCDC}=3.8V, DCEN=1, VC[1:0]=10B, I_{LOAD}=55mA	—	—	200	mV
		—	V_{IN} =2.8V, V_{OUT_DCDC}=5.5V, DCEN=1, VC[1:0]=11B, I_{LOAD}=55mA	—	—	200	mV
I_{OCDET}	Current Overload Detection	—	V_{IN} =2.8V, V_{OUT_DCDC}=3.8V, VC[1:0]=01B	Typ. -5%	90	Typ. +5%	mA
		—	V_{IN} =2.8V, V_{OUT_DCDC}=3.8V, VC[1:0]=10B	Typ. -5%	100	Typ. +5%	mA
		—	V_{IN} =2.8V, V_{OUT_DCDC}=5.5V, VC[1:0]=11B	Typ. -5%	120	Typ. +5%	mA
t_{OCDET}	Current Overload Detection Time	—	V_{IN} =2.8V, V_{OUT_DCDC}=3.8V, VC[1:0]=01B, I_{LOAD} from 35mA to 150mA	—	—	1400	μs
		—	V_{IN} =2.8V, V_{OUT_DCDC}=3.8V, VC[1:0]=10B, I_{LOAD} from 55mA to 150mA	—	—	1400	μs
		—	V_{IN} =2.8V, V_{OUT_DCDC}=5.5V, VC[1:0]=11B, I_{LOAD} from 55mA to 150mA	—	—	1400	μs
t_{OFF}	LDO Output Turn Off Time	—	V_{IN} =2.8V, V_{OUT_DCDC}=3.8V, VC[1:0]=01B, I_{LOAD}=35mA, V_{OUT_LDO} from V_{OUT_LDO} to 0.4V	—	—	750	μs
		—	V_{IN} =2.8V, V_{OUT_DCDC}=3.8V, VC[1:0]=10B, I_{LOAD}=55mA, V_{OUT_LDO} from V_{OUT_LDO} to 0.4V	—	—	750	μs
		—	V_{IN} =2.8V, V_{OUT_DCDC}=5.5V, VC[1:0]=11B, I_{LOAD}=55mA, V_{OUT_LDO} from V_{OUT_LDO} to 0.4V	—	—	750	μs

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{ON}	LDO Output Turn on Time	—	V_{IN}=2.8V, V_{OUT_DCDC}=3.8V, VC[1:0]=01B, I_{LOAD}=35mA, V_{OUT_LDO} from 0V to 1.66V	—	—	750	μs
		—	V_{IN} =2.8V, V_{OUT_DCDC}=3.8V, VC[1:0]=10B, I_{LOAD}=55mA, V_{OUT_LDO} from 0V to 2.76V	—	—	750	μs
		—	V_{IN} =2.8V, V_{OUT_DCDC}=5.5V, VC[1:0]=11B, I_{LOAD}=55mA, V_{OUT_LDO} from 0V to 4.6V	—	—	750	μs
$V5_{PWRGOOD}$	LDO 5V Power Good Voltage	—	V_{IN}=2.8V, V_{OUT_DCDC}=5.5V, no load	V_{OUT_LDO} ×94%	—	—	V
$V3_{PWRGOOD}$	LDO 3V Power Good Voltage	—	V_{IN}=2.8V, V_{OUT_DCDC}=3.8V, no load		—	—	V
$V18_{PWRGOOD}$	LDO 1.8V Power Good Voltage	—	V_{IN}=2.8V, V_{OUT_DCDC}=3.8V, no load		—	—	V

ISO7816-3 Interface Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IH}	Smart Card Interface Input High Voltage for CIO, CC4, CC8, CDET	—	—	$0.6V_{CRDVCC}$	—	V_{CRDVCC}	V
V_{IL}	Smart Card Interface Input Low Voltage for CIO, CC4, CC8, CDET	—	—	0	—	$0.2V_{CRDVCC}$	V
V_{OH}	Smart Card Interface Output High Voltage for CIO, CC4, CC8, CCLK, CRST	—	-0.1mA < I_{OH} < 0	$0.8V_{CRDVCC}$	—	V_{CRDVCC}	V
V_{OL}	Smart Card Interface Output Low Voltage for CIO, CC4, CC8, CCLK, CRST	—	0 < I_{OL} < 1mA	0	—	$0.15V_{CRDVCC}$	V
t_R, t_F	Smart Card Interface Rising and Falling Slew Rate	—	CIO, CC4, CC8 pins, C_{IN} (ICC)=30pF max.	—	0	1	μs
		—	CCLK pin, C_{IN} (ICC)=30pF max.	—	0	1	μs
		—	CRST pin, C_{IN} (ICC)=30pF max.	—	0	1	μs
R_{PH}	Pull-high Resistance for CDET Pin	3V	—	20	60	100	kΩ
		5V		10	30	50	
	Pull-high Resistance for CIO Pin	—	—	7.5	15	22.5	kΩ

LXT Oscillator Electrical Characteristics

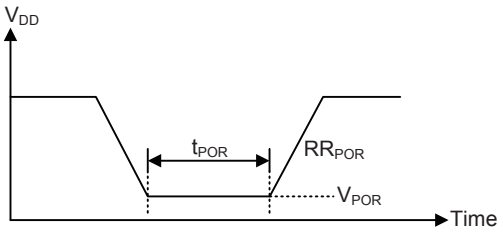
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	External R=10MΩ, C1/C2=20pF	2.2	—	5.5	V
I_{LXT}	Operating Current	3V	External R=10MΩ, C1/C2=20pF, C_L=12.5pF	—	0.9	1.5	μA
C_L	Load Capacitor	—	—	6.0	12.5	—	pF
ESR	Equivalent Series Resistance	—	—	—	—	40	kΩ
OA	Oscillation Allowance or Negative Resistance	—	External R=10MΩ, C1/C2=20pF	200	—	—	kΩ

Power-on Reset Characteristics

Ta=25°C

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



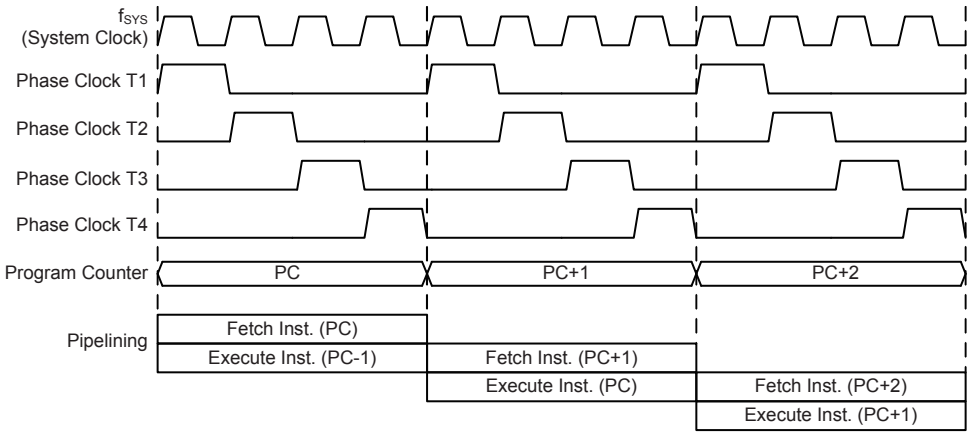
System Architecture

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

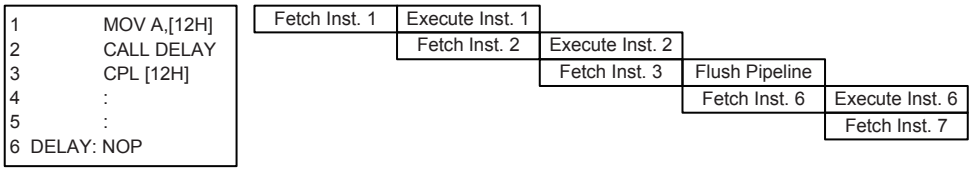
Clocking and Pipelining

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

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 Featching

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. As these devices memory capacity are greater than 8K words, the Program Memory address may be located in a certain program memory bank which is selected by the program memory bank pointer bit, PBPn. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

Device	Program Counter	
	High Byte	Low Byte (PCL)
HT66F4360	PBP0, PC12~PC8	PC7~PC0
HT66F4370	PBP1~PBP0, PC12~PC8	PC7~PC0
HT66F4390	PBP2~PBP0, PC12~PC8	PC7~PC0

Program Counter

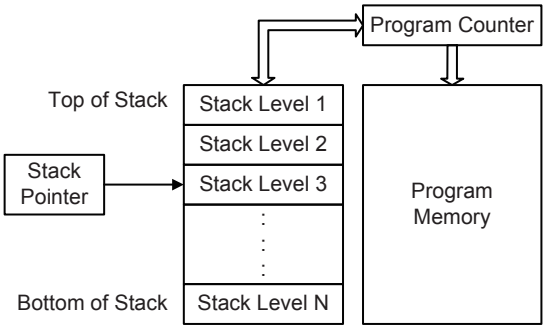
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

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

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

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



Note: N=12 for HT66F4360/HT66F4370, N=16 for HT66F4390

Arithmetic and Logic Unit – ALU

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

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

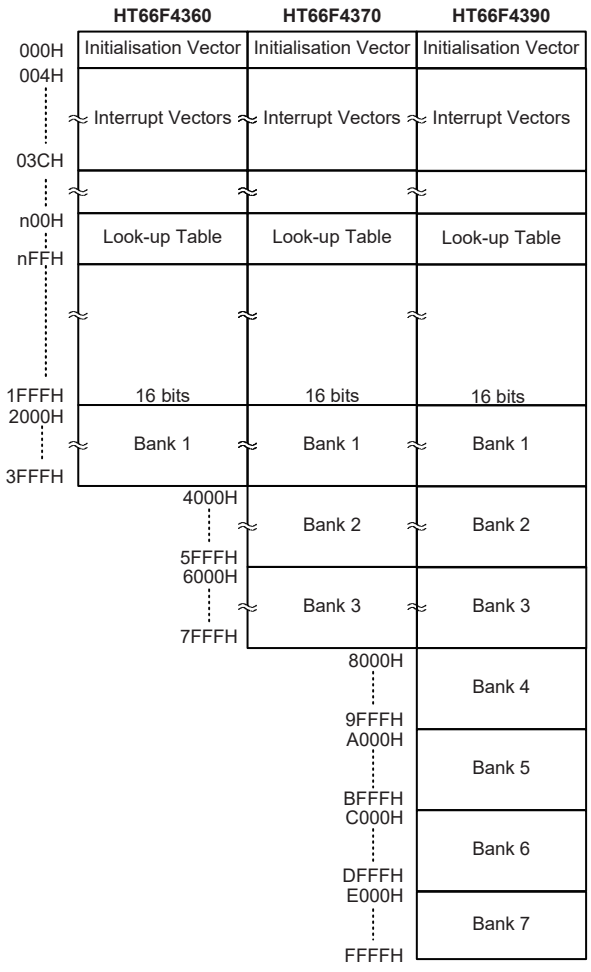
Flash Program Memory

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

Device	Capacity	Banks
HT66F4360	16K × 16	0~1
HT66F4370	32K × 16	0~3
HT66F4390	64K × 16	0~7

Structure

The Program Memory has a capacity of 16K×16 to 64K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer registers.



Program Memory Structure

Special Vectors

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

Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the "TABRD [m]" or "TABRDL [m]" instructions 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.

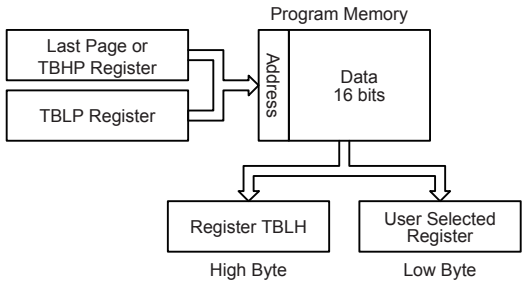


Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is "7F00H" which refers to the start address of the last page within the 32K Program Memory of the HT66F4370 if the ISP bootloader provided by HOLTEK IDE tool is not used. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "7F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page pointed by the TBHP register if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

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

Table Read Program Example

```
rombank 3 code3
ds .section 'data'
tempreg1 db  ?          ; temporary register#1
tempreg2 db  ?          ; temporary register#2
:
:
code0 .section 'code'
mov  a,06h              ; initialise table pointer - note that this address is referenced
mov  tblp,a             ; to the last page or the page that tbhp pointed
mov  a,7fh              ; initialise high table pointer
mov  tbhp,a             ; it is not necessary to set tbhp if executing tabrdl
:
:
tabrdc tempreg1         ; transfers value in table referenced by table pointer
                        ; register pair to tempregl
tabrdl tempreg1         ; data at prog.memory address 7F06H transferred to tempreg1 and TBLH
dec  tblp               ; reduce value of table pointer by one
tabrdc tempreg2         ; transfers value in table referenced by table pointer
                        ; register pairto tempreg2
tabrdl tempreg2         ; data at prog.memory address 7F05H transferred to tempreg2 and TBLH
                        ; in this example the data 1AH is transferred to tempreg1 and data
                        ; 0FH to tempreg2
                        ; the value 00H will be transferred to the high byte register TBLH
        :
        :
Code3 .section  'code'
org  1F00h              ; sets initial address of lastpage
dc   00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

In Circuit Programming – ICP

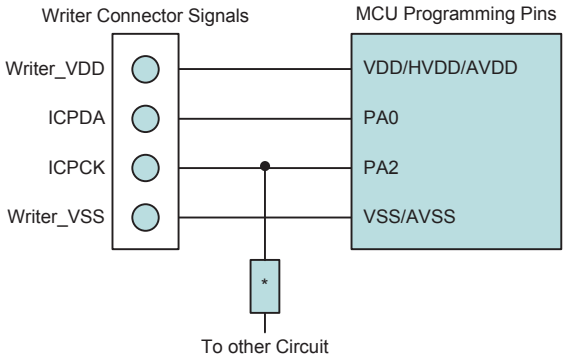
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VDD	VDD/HVDD/AVDD	Power Supply
VSS	VSS/AVSS	Ground

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

During the programming process, 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 300Ω or the capacitance of * must be less than 1nF.

On-Chip Debug Support – OCDS

There is an EV chip named HT66V43x0 which is used to emulate the real MCU device named HT66F43x0. The EV chip device also provides the "On-Chip Debug" function to debug the real MCU device during development process. The EV chip and real MCU devices, HT66V43x0 and HT66F43x0, are almost functional compatible except the "On-Chip Debug" function. Users can use the EV chip device to emulate the real MCU device behaviors by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip device for debugging, the corresponding pin functions shared with the OCDSDA and OCDSCK pins in the real MCU device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

Holtek e-Link Pins	EV Chip OCDS Pins	Pin Description
OCDSDA	OCDSDA	On-Chip Debug Support Data/Address input/output
OCDSCK	OCDSCK	On-Chip Debug Support Clock input
VDD	VDD/HVDD/AVDD	Power Supply
VSS	VSS/AVSS	Ground

In Application Programming – IAP

Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. The provision of not only an IAP function but also an additional ISP function offers users the convenience of Flash Memory multi-programming features. The convenience of the IAP function is that it can execute the updated program procedure using its internal firmware, without requiring an external Program Writer or PC. In addition, the IAP interface can also be any type of communication protocol, such as UART or USB, using I/O pins. Regarding the internal firmware, the user can select versions provided by HOLTEK or create their own. The following section illustrates the procedures regarding how to implement the IAP firmware.

Flash Memory Read/Write Size

The flash memory Erase and Write operations are carried out in a page format while the Read operation is carried out in a word format. The page size and write buffer size are both assigned with a capacity of 64 or 128 words respectively. Note that the Erase operation should be executed before the Write operation is executed.

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

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

Device	Program Memory Size	Erase	Write	Read	Page
HT66F4360	16K×16	64 words/page	64 words/time	1 word/time	64 words
HT66F4370	32K×16				
HT66F4390	64K×16	128 words/page	128 words/page	1 word/time	128 words

IAP Operation Format

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

"x": don't care

HT66F4360 Erase Page Number and Selection

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

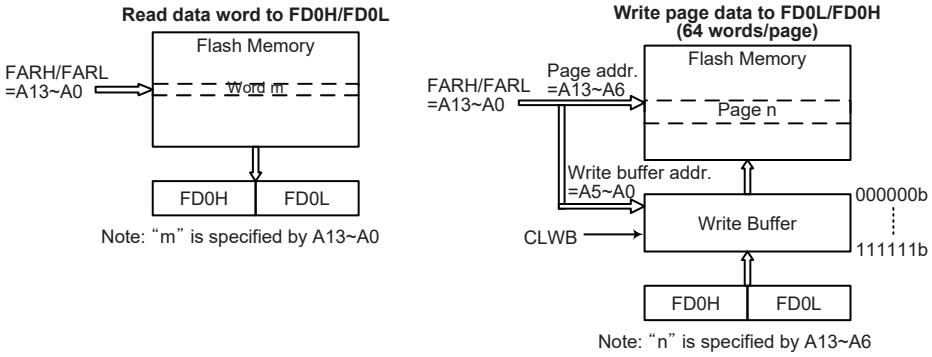
"x": don't care

HT66F4370 Erase Page Number and Selection

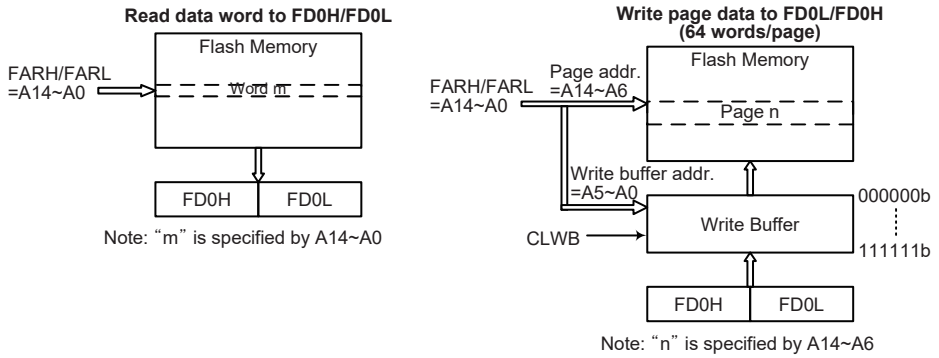
Erase Page	FARH	FARL [7]	FARL [6:0]
0	0000 0000	00	xx xxxx
1	0000 0000	01	xx xxxx
2	0000 0000	10	xx xxxx
3	0000 0000	11	xx xxxx
4	0000 0001	00	xx xxxx
: :	: :	: :	: :
510	1111 1111	10	xx xxxx
511	1111 1111	11	xx xxxx

"x": don't care

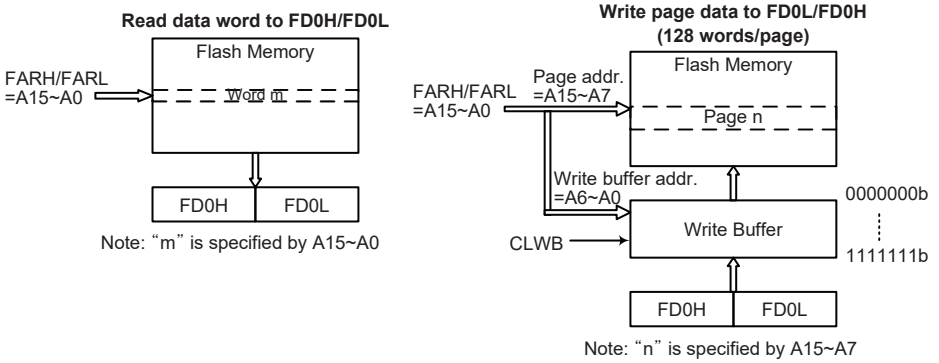
HT66F4390 Erase Page Number and Selection



HT66F4360 Flash Memory IAP Read/Write Structure



HT66F4370 Flash Memory IAP Read/Write Structure



HT66F4390 Flash Memory IAP Read/Write Structure

- **Write Buffer**

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

  The write buffer size is 64 or 128 words corresponding to a page respectively. The write buffer address is mapped to a specific flash memory page specified by the memory address bits, A13~A6, A14~A6 or A15~A7. The data written into the FD0L and FD0H registers will be loaded into the write buffer. When data is written into the high byte data register, FD0H, it will result in the data stored in the high and low byte data registers both being written into the write buffer. It will also cause the flash memory address to be incremented by one, after which the new address will be loaded into the FARH and FARL address registers. When the flash memory address reaches the page boundary, 111111b of a page with 64 words or 1111111b of a page with 128 words, the address will now not be incremented but will stop at the last address of the page. At this point a new page address should be specified for any other erase/write operations.

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

IAP Flash Program Memory Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
FCR	CFWEN	FMOD2	FMOD1	FMOD0	BWT	FWT	FRDEN	FRD
FRCR	D7	D6	—	D4	—	—	—	CLWB
FARL	A7	A6	A5	A4	A3	A2	A1	A0
FARH (HT66F4360)	—	—	A13	A12	A11	A10	A9	A8
FARH (HT66F4370)	—	A14	A13	A12	A11	A10	A9	A8
FARH (HT66F4390)	A15	A14	A13	A12	A11	A10	A9	A8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Registers List

- **FARL Register**

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

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

- **FARH Register – HT66F4360**

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

Bit 7~6 Unimplemented, read as "0"

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

- **FARH Register – HT66F4370**

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

Bit 7 Unimplemented, read as "0"

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

- **FARH Register – HT66F4390**

Bit	7	6	5	4	3	2	1	0
Name	A15	A14	A13	A12	A11	A10	A9	A8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 Flash Memory Address bit 15 ~ bit 8

- **FD0L Register**

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

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

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

- **FD0H Register**

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

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

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

- **FD1L Register**

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

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

- **FD1H Register**

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

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

- **FD2L Register**

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

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

- **FD2H Register**

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

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

- **FD3L Register**

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

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

- **FD3H Register**

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

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

- **FCR Register**

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

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

Bit 6~4 **FMOD2~FMOD0**: Flash Memory Mode selection
000: Write Mode
001: Page erase Mode
010: Reserved
011: Reserved
101: Reserved
110: Flash memory Erase/Write function Enable Mode
111: Reserved

These bits are used to select the Flash Memory operation modes. Note that the "Flash memory Erase/Write function Enable Mode" should first be successfully enabled before the Erase or Write Flash memory operation is executed.

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

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

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

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

Note: The FWT, FRDEN and FRD bits cannot be set to "1" at the same time with a single instruction.

- **FRCR Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	—	D4	—	—	—	CLWB
R/W	R/W	R/W	—	R/W	—	—	—	R/W
POR	0	0	—	0	—	—	—	0

Bit 7~6 **D7~D6**: Reserved, can not be used

Bit 5 Unimplemented, read as "0"

Bit 4 **D4**: Reserved, can not be used

Bit 3~1 Unimplemented, read as "0"

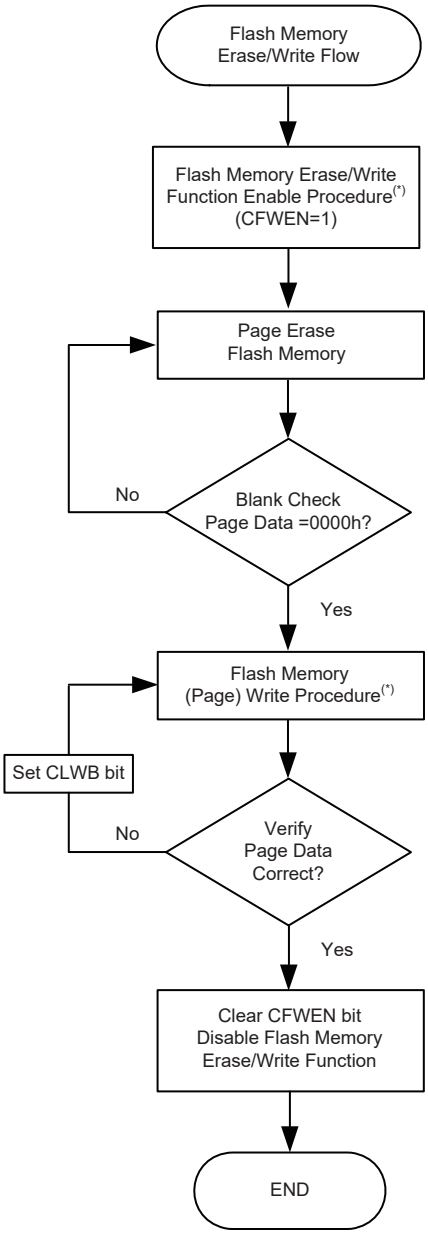
Bit 0 **CLWB**: Flash memory Write Buffer Clear control
0: Do not initiate Write Buffer Clear process or Write Buffer Clear process is completed
1: Initiate Write Buffer Clear process
This bit is set by software and cleared by hardware when the Write Buffer Clear process is completed.

Flash Memory Erase/Write Flow

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

- **Flash Memory Erase/Write Flow Descriptions**

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



Flash Memory Erase/Write Flow

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

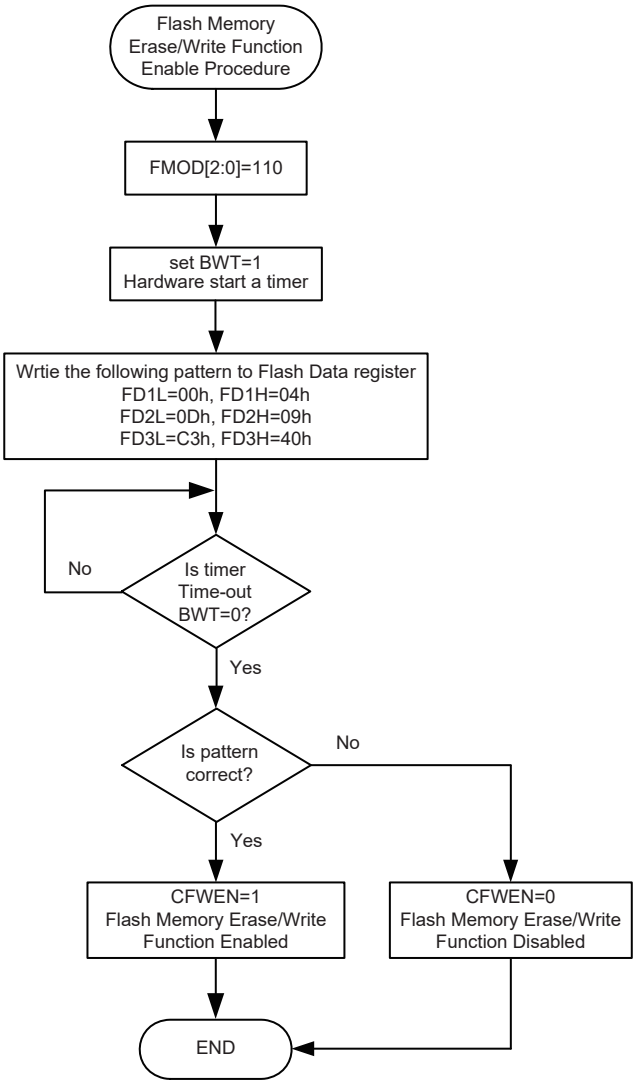
Flash Memory Erase/Write Function Enable Procedure

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

- **Flash Memory Erase/Write Function Enable Procedure Description**

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

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



Flash Memory Erase/Write Function Enable Procedure

Flash Memory Write Procedure

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

The write buffer size is 64 or 128 words respectively, known as a page, whose address is mapped to a specific flash memory page specified by the memory address bits, A13~A6, A14~A6 or A15~A7. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits specify.

- **Flash Memory Consecutive Write Description**

The maximum amount of write data is 64 or 128 words respectively for each write operation. The write buffer address will be automatically incremented by one when consecutive write operations are executed. The start address of a specific page should first be written into the FARL and FARH registers. Then the data word should first be written into the FD0L register and then the FD0H register. At the same time the write buffer address will be incremented by one and then the next data word can be written into the FD0L and FD0H registers for the next address without modifying the address register pair, FARH and FARL. When the write buffer address reaches the page boundary the address will not be further incremented but will stop at the last address of the page.

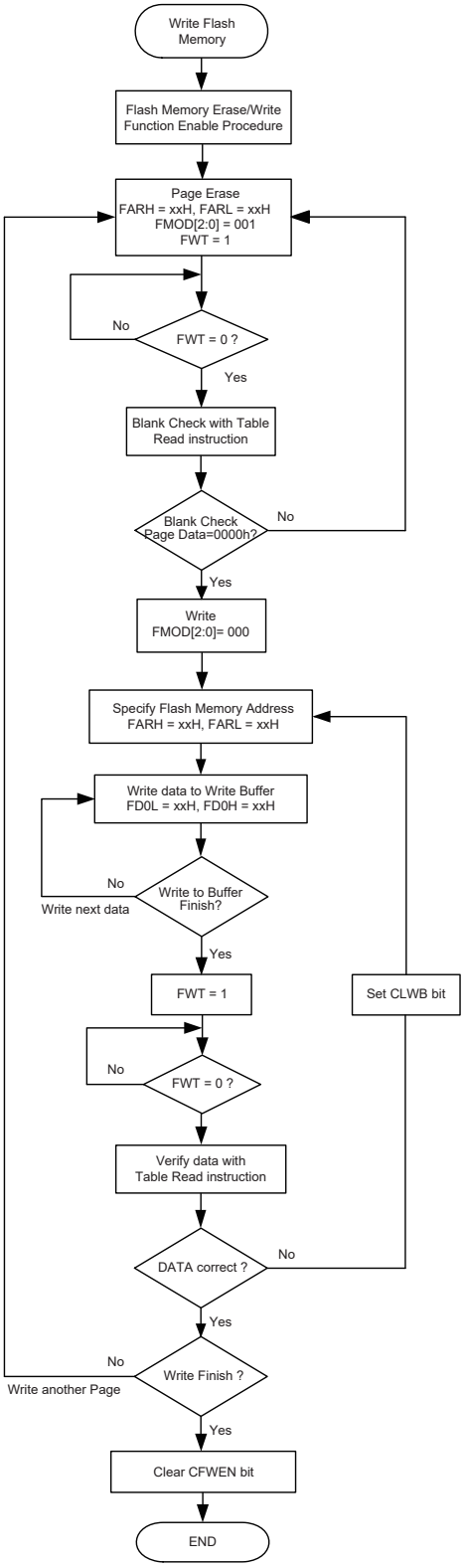
1. Activate the "Flash Memory Erase/Write function enable procedure". Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the "Flash Memory Erase/Write function enable procedure" for more details.
2. Set the FMOD field to "001" to select the erase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

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

   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to "000" to select the write operation.
5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L and FD0H registers within a page as specified by their consecutive addresses. The maximum written data number is 64 or 128 words respectively.
6. Set the FWT bit high to write the data words from the write buffer to the flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

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

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



Flash Memory Consecutive Write Procedure

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

- **Flash Memory Non-Consecutive Write Description**

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

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

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

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

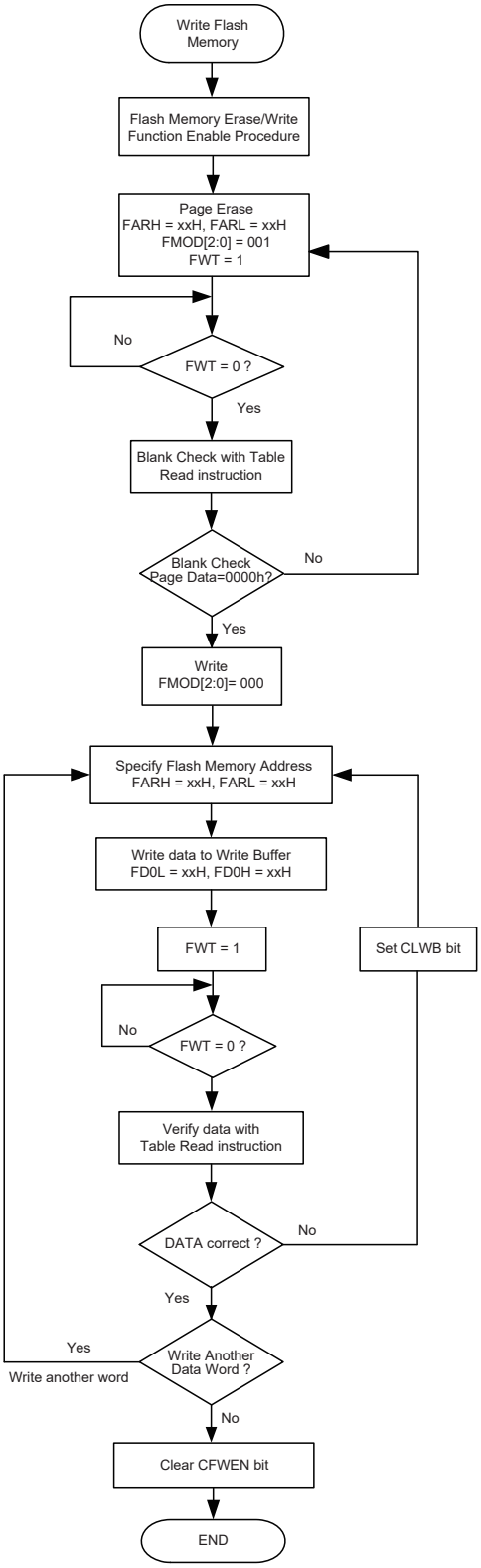
     Go to step 4 if the erase operation is successful.
  4. Set the FMOD field to "000" to select the write operation.
  5. Setup the desired address ADDR1 in the FARH and FRARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
  6. Set the FWT bit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

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

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

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

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



Flash Memory Non-Consecutive Write Procedure

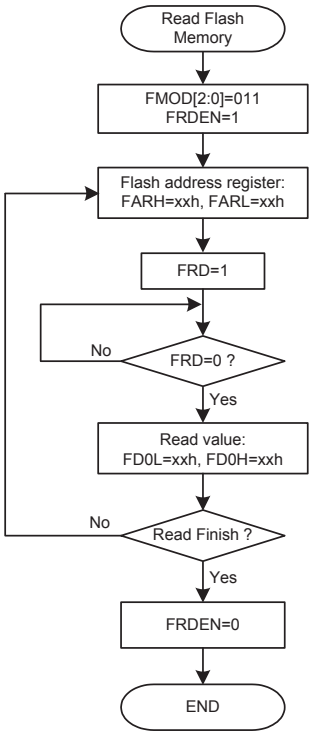
Note: When the FWT bit is set to 1 all CPU operations will temporarily cease.

- **Important Points to Note for Flash Memory Write Operations**

1. The "Flash Memory Erase/Write Function Enable Procedure" must be successfully activated before the Flash Memory erase/write operation is executed.
2. The Flash Memory erase operation is executed to erase a whole page.
3. The whole write buffer data will be written into the flash memory in a page format. The corresponding address cannot exceed the page boundary.
4. Bit 7 ~ bit 1 in the FRCR register must remain at "0" to avoid unpredictable errors during the IAP supported operations.
5. After the data is written into the flash memory the flash memory contents must be read out using the table read instruction, TABRD, and checked if it is correct or not. If the data written into the flash memory is incorrect, the write buffer should be cleared by setting the CLWB bit high and then writing the data again into the write buffer. Then activate a write operation on the same flash memory page without erasing it. The data check, buffer clear and data re-write steps should be repeatedly executed until the data written into the flash memory is correct.
6. The system frequency should be setup to the maximum application frequency when data write and data check operations are executed using the IAP function.

Flash Memory Read Procedure

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



Flash Memory Read Procedure

Note: When the FRD bit is set to "1" all CPU operations will temporarily cease.

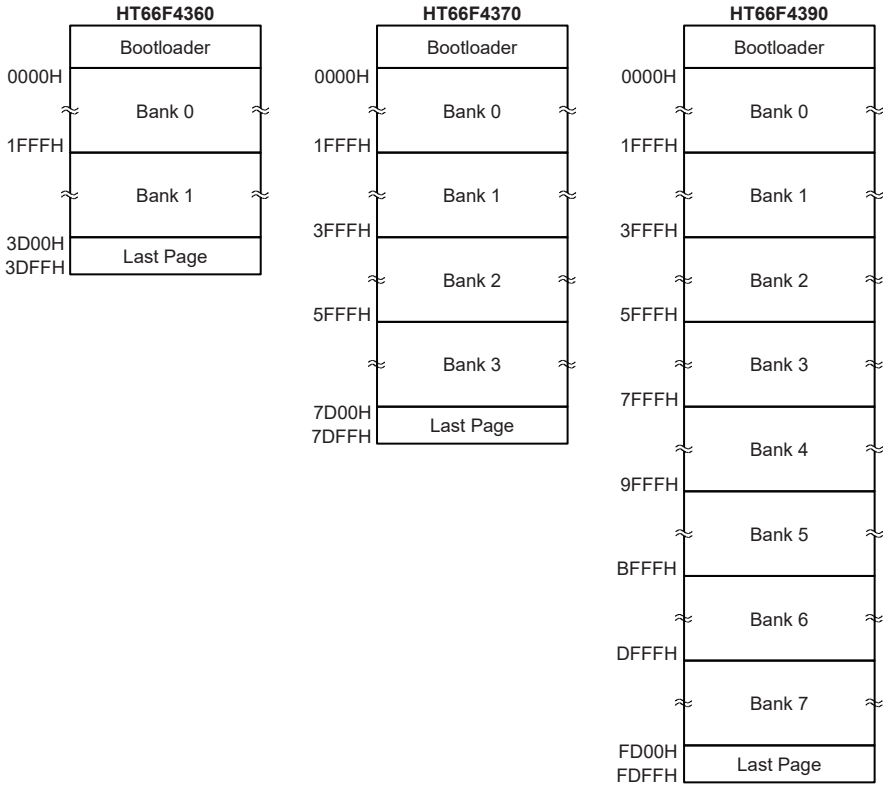
In System Programming – ISP

As an additional convenience, Holtek has provided a means of programming the microcontroller in-system using a two-line USB 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 microcontroller device.

The Program Memory can be programmed serially in-system using the USB interface, namely using the UDN and UDP pins. The power is supplied by the UBUS pin. The technical details regarding the in-system programming are beyond the scope of this document and will be supplied in supplementary literature. The Flash Program Memory Read/Write function is implemented using a series of registers.

ISP Bootloader

An ISP Bootloader function is provided to upgrade the software in the Flash memory. The user can utilise either the ISP Bootloader application software provided by the Holtek IDE tools or to create their own Bootloader software. When the Holtek Bootloader software is selected note that it will occupy an area of 0.5K capacity area in the Flash memory. The accompanying diagram illustrates the Flash memory structure including the Holtek Bootloader software.



Flash Program Memory Structure including Bootloader

Data Memory

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

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

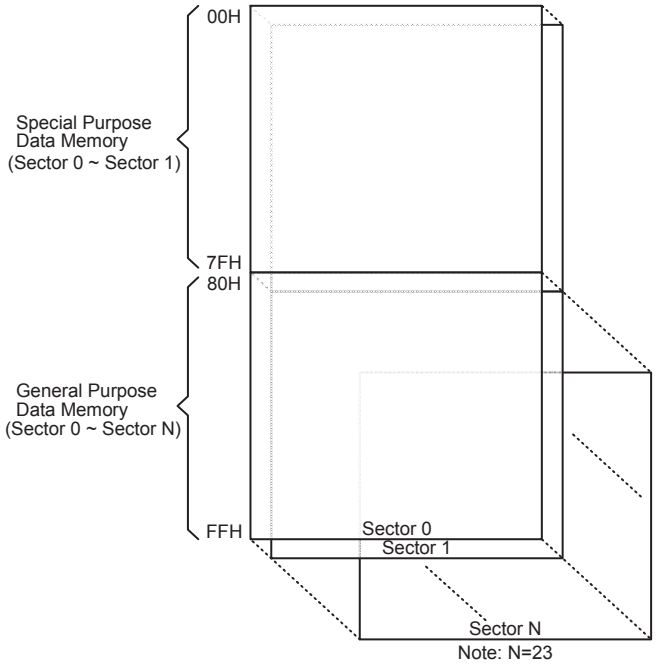
Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value.

Structure

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

Device	Special Purpose Data Memory	General Purpose Data Memory	
	Located Sectors	Capacity	Sector: Address
HT66F4360 HT66F4370 HT66F4390	0, 1	3072 × 8	0: 80H~FFH 1: 80H~FFH ⋮ 22: 80H~FFH 23: 80H~FFH

Data Memory Summary



Data Memory Structure

Data Memory Addressing

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

Direct Addressing can be used in all sectors using the corresponding instruction which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address "m" in the extended instructions can be 13 bits for this series of devices, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH	PBP	
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	INTC0	
10H	INTC1	
11H	INTC2	
12H	INTC3	
13H	MFI0	
14H	MFI1	
15H	MFI2	
16H	MFI3	
17H	INTEG	
18H	PA	PAS0
19H	PAC	PAS1
1AH	PAPU	PBS0
1BH	PAWU	PBS1
1CH	PB	PCS0
1DH	PBC	PCS1
1EH	PBPU	PES0
1FH	PC	PES1

	Sector 0	Sector 1
20H	PCC	
21H	PCPU	
22H	PE	PF
23H	PEC	PFC
24H	PEPU	PFPU
25H	SCC	PFS0
26H	HIRCC	PFS1
27H	HXTC	
28H	LXTC	
29H	LVDC	
2AH	LVRC	
2BH	WDTC	
2CH	RSTC	
2DH	RSTFC	
2EH	PLLC	
2FH	PSCR	
30H	TB0C	
31H	TB1C	
32H	USB_STAT	IICC0
33H	UINT	IICC1
34H	USC	IICD
35H	UESR	IICA
36H	UCC	IICTOC
37H	AWR	SADOL
38H	STLI	SADOH
39H	STLO	SADC0
3AH	SIES	SADC1
3BH	MISC	CMP0C
3CH	UFIEN	CMP0VOS
3DH	UFOEN	CMP1C
3EH	UFC0	CMP1VOS
3FH	UFC1	

	Sector 0	Sector 1
40H	FIFO0	
41H	FIFO1	
42H	FIFO2	
43H	FIFO3	
44H	FIFO4	
45H	FIFO5	
46H	FIFO6	
47H	FIFO7	
48H		FRCR
49H		FCR
4AH	FARL	FARH
4BH	FD0L	FD0H
4CH	FD1L	FD1H
4DH	FD2L	FD2H
4EH	FD3L	FD3H
4FH	DC2DC	
50H	CCR	
51H	CSR	
52H	CCCR	
53H	CETU1	
54H	CETU0	
55H	CGT1	
56H	CGT0	
57H	CWT2	
58H	CWT1	
59H	CWT0	
5AH	CIER	
5BH	CIPR	
5CH	CTXB	
5DH	CRXB	
5EH	STMC0	CTM0C0
5FH	STMC1	CTM0C1

	Sector 0	Sector 1
60H	STMDL	CTM0DL
61H	STMDH	CTM0DH
62H	STMAL	CTM0AL
63H	STMAH	CTM0AH
64H	STMRP	CTM1C0
65H	PTMC0	CTM1C1
66H	PTMC1	CTM1DL
67H	PTMDL	CTM1DH
68H	PTMDH	CTM1AL
69H	PTMAL	CTM1AH
6AH	PTMAH	
6BH	PTMRPL	
6CH	PTMRPH	
6DH	SPI0C0	
6EH	SPI0C1	
6FH	SPI0D	
70H	SPI1C0	
71H	SPI1C1	
72H	SPI1D	
73H	U0SR	
74H	U0CR1	
75H	U0CR2	
76H	TXR_RXR0	
77H	BRG0	
78H	U1SR	
79H	U1CR1	
7AH	U1CR2	
7BH	TXR_RXR1	
7CH	BRG1	
7DH	URFC	
7EH	ISOC	
7FH	SYSC	

: Unused, read as 00H

Special Purpose Data Memory Structure – HT66F4360/HT66F4370

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH	PBP	
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	INTC0	
10H	INTC1	
11H	INTC2	
12H	INTC3	
13H	MFI0	
14H	MFI1	
15H	MFI2	
16H	MFI3	
17H	INTEG	
18H	PA	PAS0
19H	PAC	PAS1
1AH	PAPU	PBS0
1BH	PAWU	PBS1
1CH	PB	PCS0
1DH	PBC	PCS1
1EH	PBPU	PES0
1FH	PC	PES1

	Sector 0	Sector 1
20H	PCC	
21H	PCPU	
22H	PE	PF
23H	PEC	PFC
24H	PEPU	PFPU
25H	SCC	PFS0
26H	HIRCC	PFS1
27H	HXTC	
28H	LXTC	
29H	LVDC	
2AH	LVRC	
2BH	WDTC	
2CH	RSTC	
2DH	RSTFC	
2EH	PLLC	
2FH	PSCR	
30H	TB0C	
31H	TB1C	
32H	USB_STAT	IICC0
33H	UINT	IICC1
34H	USC	IICD
35H	UESR	IICA
36H	UCC	IICTOC
37H	AWR	SADOL
38H	STLI	SADOH
39H	STLO	SADC0
3AH	SIES	SADC1
3BH	MISC	CMP0C
3CH	UFIEN	CMP0VOS
3DH	UFOEN	CMP1C
3EH	UFC0	CMP1VOS
3FH	UFC1	

	Sector 0	Sector 1
40H	FIFO0	EEC
41H	FIFO1	
42H	FIFO2	
43H	FIFO3	
44H	FIFO4	
45H	FIFO5	
46H	FIFO6	
47H	FIFO7	
48H	EEA	FRCR
49H	EED	FCR
4AH	FARL	FARH
4BH	FD0L	FD0H
4CH	FD1L	FD1H
4DH	FD2L	FD2H
4EH	FD3L	FD3H
4FH	DC2DC	
50H	CCR	
51H	CSR	
52H	CCCR	
53H	CETU1	
54H	CETU0	
55H	CGT1	
56H	CGT0	
57H	CWT2	
58H	CWT1	
59H	CWT0	
5AH	CIER	
5BH	CIPR	
5CH	CTXB	
5DH	CRXB	
5EH	STMC0	CTM0C0
5FH	STMC1	CTM0C1

	Sector 0	Sector 1
60H	STMDL	CTM0DL
61H	STMDH	CTM0DH
62H	STMAL	CTM0AL
63H	STMAH	CTM0AH
64H	STMRP	CTM1C0
65H	PTMC0	CTM1C1
66H	PTMC1	CTM1DL
67H	PTMDL	CTM1DH
68H	PTMDH	CTM1AL
69H	PTMAL	CTM1AH
6AH	PTMAH	
6BH	PTMRPL	
6CH	PTMRPH	
6DH	SPI0C0	
6EH	SPI0C1	
6FH	SPI0D	
70H	SPI1C0	
71H	SPI1C1	
72H	SPI1D	
73H	U0SR	
74H	U0CR1	
75H	U0CR2	
76H	TXR_RXR0	
77H	BRG0	
78H	U1SR	
79H	U1CR1	
7AH	U1CR2	
7BH	TXR_RXR1	
7CH	BRG1	
7DH	URFC	
7EH	ISOC	
7FH	SYSC	

: Unused, read as 00H

Special Purpose Data Memory Structure – HT66F4390

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, IAR2

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

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

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

Indirect Addressing Program Example

Example 1

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

Example 2

```
data    .section 'data'
adres1  db ?
adres2  db ?
adres3  db ?
adres4  db ?
block   db ?
code    .section at 0 'code'
org     00h
start:
mov  a,04h                ; setup size of block
mov  block,a
mov  a,01h                ; setup the memory sector
mov  mp1h,a
mov  a,offset adres1    ; Accumulator loaded with first RAM address
mov  mp1l,a               ; setup memory pointer with first RAM address
loop:
clr  IAR1                 ; clear the data at address defined by MP1
inc  mp1l                 ; increment memory pointer MP1L
sdz  block                ; check if last memory location has been cleared
jmp  loop
continue:
     :
```

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

Direct Addressing Program Example using extended instructions

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

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

Program Memory Bank Pointer – PBP

For the HT66F43x0 device the Program Memory is divided into several banks. Selecting the required Program Memory area is achieved using the Program Memory Bank Pointer, PBP. The PBP register should be properly configured before the device executes the "Branch" operation using the "JMP" or "CALL" instruction. After that a jump to a non-consecutive Program Memory address which is located in a certain bank selected by the program memory bank pointer bits will occur.

PBP Register – HT66F4360

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	PBP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~1 **D7~D1**: General data bits and can be read or written

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

PBP Register – HT66F4370

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	PBP1	PBP0
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 7~2 **D7~D2**: General data bits and can be read or written

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

PBP Register – HT66F4390

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	PBP2	PBP1	PBP0
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 7~3 **D7~D3**: General data bits and can be read or written

Bit 2~0 **PBP2~PBP0**: Program Memory Bank Point bit 2~ bit 0
000: Bank 0
001: Bank 1
010: Bank 2
011: Bank 3
100: Bank 4
101: Bank 5
110: Bank 6
111: Bank 7

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

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

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

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

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

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

STATUS Register

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

"x": unknown

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

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

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

For SBC/ SBCM/ LSBC/ LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation zero flag. For other instructions, the CZ flag willl not be affected.

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

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

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

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

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

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

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

EEPROM Data Memory

The HT66F4390 contains an area of internal EEPROM Data Memory. EEPROM, which stands for Electrically Erasable Programmable Read Only Memory, is by its nature a non-volatile form of re-programmable memory, with data retention even when its power supply is removed. By incorporating this kind of data memory, a whole new host of application possibilities are made available to the designer. The availability of EEPROM storage allows information such as product identification numbers, calibration values, specific user data, system setup data or other product information to be stored directly within the product microcontroller. The process of reading and writing data to the EEPROM memory has been reduced to a very trivial affair.

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 256×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and a data register in Sector 0 and a single control register in Sector 1.

EEPROM Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEA	EEA7	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
EEC	–	–	–	–	WREN	WR	RDEN	RD

EEPROM Register List

EEA Register

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

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

EED Register

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

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

EEC Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3 **WREN**: Data EEPROM Write Enable
0: Disable
1: Enable
This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2 **WR**: EEPROM Write Control
0: Write cycle has finished
1: Activate a write cycle
This is the Data EEPROM Write Control Bit and when set high by the application program will activate a write cycle. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.

Bit 1 **RDEN**: Data EEPROM Read Enable
0: Disable
1: Enable
This is the Data EEPROM Read Enable Bit which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.

Bit 0 **RD**: EEPROM Read Control
0: Read cycle has finished
1: Activate a read cycle
This is the Data EEPROM Read Control Bit and when set high by the application program will activate a read cycle. This bit will be automatically reset to zero by the hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

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

Reading Data from the EEPROM

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

Writing Data to the EEPROM

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

Write Protection

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

EEPROM Interrupt

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

Programming Considerations

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

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

Programming Examples

Reading data from the EEPROM – polling method

```
MOV  A, EEPROM_ADRES          ; user defined address
MOV  EEA, A
MOV  A, 40H                   ; setup memory pointer MP1L
MOV  MP1L, A                  ; MP1L points to EEC register
MOV  A, 01H                   ; setup memory pointer MP1H
MOV  MP1H, A
SET  IAR1.1                   ; set RDEN bit, enable read operations
SET  IAR1.0                   ; start Read Cycle - set RD bit
BACK:
SZ   IAR1.0                   ; check for read cycle end
JMP  BACK
CLR  IAR1                     ; disable EEPROM read/write
CLR  MP1H
MOV  A, EED                   ; move read data to register
MOV  READ_DATA, A
```

Writing Data to the EEPROM – polling method

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

Oscillators

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

Oscillator Overview

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

For USB applications, the HXT pins must be connected to a 6MHz or 12MHz crystal if the HXT oscillator is selected to be used.

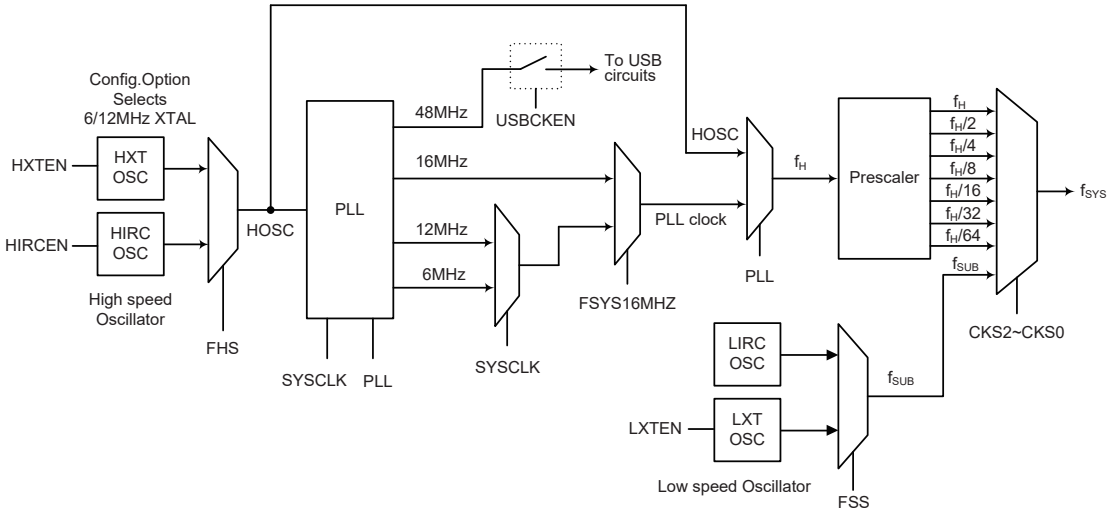
Type	Name	Frequency	Pins
External High Speed Crystal	HXT	6 MHz or 12 MHz	OSC1/OSC2
Internal High Speed RC	HIRC	12 MHz	—
External Low Speed Crystal	LXT	32.768 kHz	XT1/XT2
Internal Low Speed RC	LIRC	32 kHz	—

Oscillator Types

System Clock Configurations

There are four methods of generating the system clock, two high speed oscillators for all devices and two low speed oscillators. The high speed oscillator is the external crystal/ceramic oscillator, HXT, and the internal 12 MHz RC oscillator, HIRC. The two low speed oscillators are the internal 32 kHz RC oscillator, LIRC, and the external 32.768 kHz crystal oscillator, LXT. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

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

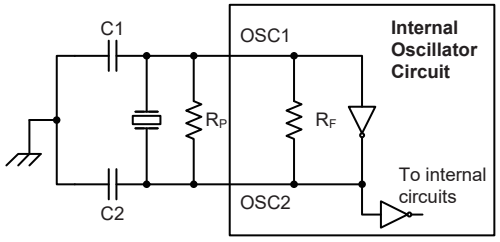


System Clock Configurations

External Crystal/Ceramic Oscillator – HXT

The External Crystal/Ceramic System Oscillator is one of the high frequency oscillators. For most crystal oscillator configurations, the simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, without requiring external capacitors. However, for some crystal types and frequencies, to ensure oscillation, it may be necessary to add two small value capacitors, C1 and C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. For USB applications, the HXT pins must be connected to a 6MHz or 12MHz crystal if the HXT oscillator is selected to be used.

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



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

Crystal/Resonator Oscillator

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

Crystal Recommended Capacitor Values

Internal PLL Frequency Generator

The internal PLL frequency generator is used to generate the frequency for the USB interface and the system clock. This PLL generator can be enabled or disabled by the PLL control bit in the USC register. After a power on reset, the PLL control bit will be set to "0" to turn on the PLL generator. The PLL generator will provide the fixed 48MHz frequency for the USB operating frequency and another frequency for the system clock source which can be 6MHz, 12MHz or 16MHz. The selection of this system frequency is implemented using the SYSCLK, FSYS16MHZ and USBCKEN bits in the UCC register. In addition, the system clock can be selected as the HXT via these control bits. The CLK_ADJ bit is used to adjust the PLL clock automatically.

The following table illustrates the PLL output frequency selected by the related control bits.

PLL	USBCKEN	FSYS16MHZ	f_H
0	0	0	HOSC – HXT or HIRC
0	0	1	f_{PLL}=16MHz
0	1	0	f_{PLL}=6MHz or 12MHz, depending upon the "SYSCLK" bit in the UCC register.
0	1	1	f_{PLL}=16MHz
1	x	x	HOSC – HXT or HIRC

"x": don't care

SYSC Register

Bit	7	6	5	4	3	2	1	0
Name	CLK_ADJ	USBDIS	RUBUS	D4	D3	D2	IOVC1	IOVC0
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 **CLK_ADJ**: PLL clock automatic adjustment function control
0: Disable
1: Enable
Note that if the user selects the HIRC as the system clock, the CLK_ADJ bit must be set to 1 to adjust the PLL frequency automatically.

Bit 6 **USBDIS**: USB engine SIE control
Discribed elsewhere.

Bit 5 **RUBUS**: UBUS pin pull low function control
Discribed elsewhere.

Bit 4~2 **D4~D2**: Reserved data bits, should be kept low and cannot be modified.

Bit 1~0 **IOVC1~IOVC0**: Port A and Port B pins supply power select
Discribed elsewhere.

UCC Register

Bit	7	6	5	4	3	2	1	0
Name	RCTRL	SYSCLK	FSYS16MHZ	SUSP2	USBCKEN	EPS2	EPS1	EPS0
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 **RCTRL**: 7.5kΩ resistor between UDP and UBUS connection control
Discribed elsewhere.

Bit 6 **SYSCLK**: System clock frequency select
0: 12MHz
1: 6MHz
This bit is used to specify the system clock oscillator used by the MCU. If a 6MHz crystal or resonator is used for the device, this bit should be set to 1. If a 12MHz crystal or resonator is used, then this bit should be set to 0. If the 12MHz HIRC is selected, then this bit must be set to 0.

Bit 5 **FSYS16MHZ**: System clock frequency select
0: System clock comes from HXT or HIRC
1: System clock comes from PLL output – 16MHz

Bit 4 **SUSP2**: Suspend mode power consumption reduction control
Discribed elsewhere.

Bit 3 **USBCKEN**: USB clock enable control
0: Disable
1: Enable

Bit 2~0 **EPS2~EPS0**: Endpoint FIFO access selection
Discribed elsewhere.

USC Register

Bit	7	6	5	4	3	2	1	0
Name	URD	SELPS2	PLL	SELUSB	RESUME	URST	RMWK	SUSP
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R
POR	1	0	0	0	0	0	0	0

Bit 7 **URD**: USB reset signal control
Discribed elsewhere.

Bit 6 **SELPS2**: PS2 mode selection
Discribed elsewhere.

Bit 5 **PLL**: PLL circuit enable control
0: Enable the PLL
1: Disable the PLL

Bit 4 **SELUSB**: USB analog function and V33O enable control
Discribed elsewhere.

Bit 3 **RESUME**: USB Resume indication
Discribed elsewhere.

Bit 2 **URST**: USB Reset indication
Discribed elsewhere.

Bit 1 **RMWK**: USB Remote wake-up control
Discribed elsewhere.

Bit 0 **SUSP**: USB Suspend mode indication
Discribed elsewhere.

Internal High Speed RC Oscillator – HIRC

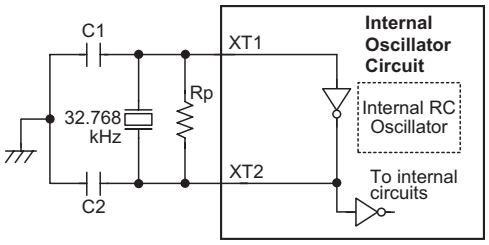
The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 12MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 3.3V or 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 12MHz will have a tolerance within 3% for non-USB mode. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins are free for use as normal I/O or other pin-shared pins. The HIRC oscillator has its own power supply pin, HVDD. The HVDD pin must be connected to VDD pin and a 0.1μF capacitor to ground.

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.

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

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



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

External LXT Oscillator

Internal 32kHz Oscillator – LIRC

The Internal 32 kHz 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 32 kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 5V and at a temperature of 25˚C degrees, the fixed oscillation frequency of 32 kHz will have a tolerance within 10%.

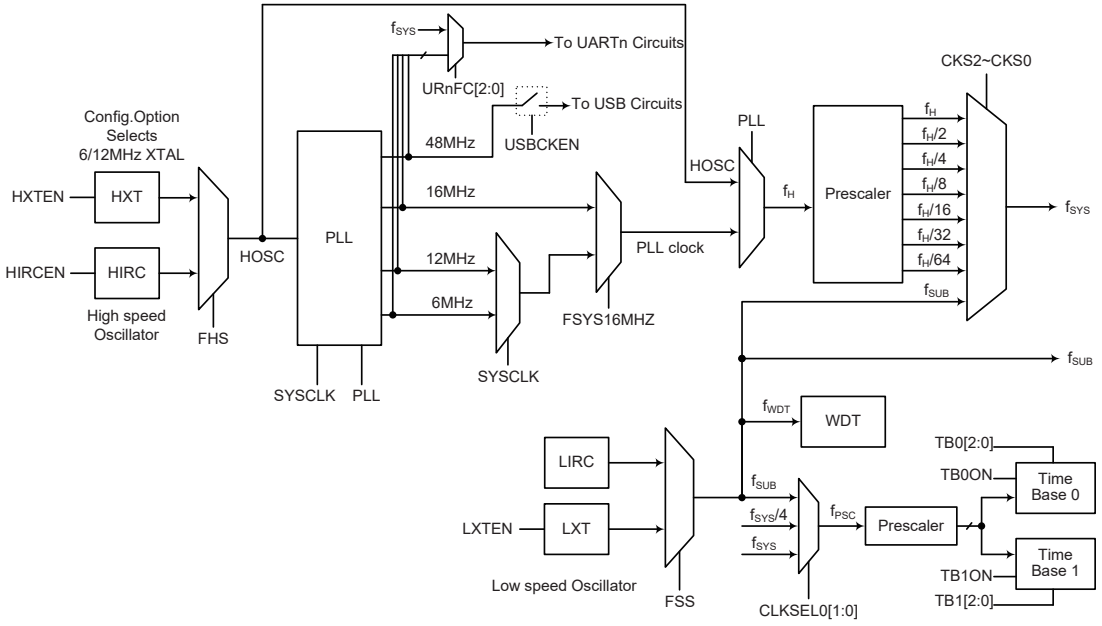
Operating Modes and System Clocks

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

System Clocks

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

The main system clock, can come from either a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from an HXT or HIRC oscillator or the USB PLL output clock, selected via configuring the FHS bit in the SCC register together with the PLL, USBCKEN and FSYS16MHZ control bits. The low speed system clock source can be sourced from the internal clock f_{SUB}. If f_{SUB} is selected then it can be sourced by either the LXT or LIRC oscillators, selected via configuring the FSS bit in the SCC register. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.



Device Clock Configurations

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

System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the NORMAL 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_{WDT}	f_{PLL}
		FHIDEN	FSIDEN	CKS2~CKS0					
NORMAL	On	x	x	000~110	f_H~f_H/64	On	On	On	On
SLOW	On	x	x	111	f_{SUB}	On/Off (1)	On	On	On/Off (3)
IDLE0	Off	0	1	000~110	Off	Off	On	On	Off
				111	On				
IDLE1	Off	1	1	xxx	On	On	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off	On/Off (2)	On
				111	Off				
SLEEP	Off	0	0	xxx	Off	Off	Off	On/Off (2)	Off

"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_{WDT} clock can be switched on or off which is controlled by the WDT function being enabled or disabled.
3. The f_{PLL} clock can be switched on or off which is controlled by the PLL bit in the USC register in any operating modes.
4. The USB function is inactive in IDLE0 and SLEEP mode.

NORMAL Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from one of the high speed oscillators, either the HXT or HIRC oscillators. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register.Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current. In the USB mode, the PLL circuit of which the clock source can be derived from the HXT or HIRC oscillator can generate a clock with a frequency of 6MHz, 12MHz or 16MHz as the device system clock.

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 = instruction is executed and when the FHIDEN and FSIDEN bit are low. In the SLEEP mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral function will also be stopped, too. However the f_{WDT} clock can continues to operate if the WDT function is always enabled.

IDLE0 Mode

The IDLE0 Mode is entered when = 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 = 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 = instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is low. In the IDLE2 Mode the CPU will be switched off but the high speed oscillator will be turned on to provide a clock source to keep some peripheral functions operational.

Control Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	FHS	FSS	FHIDEN	FSIDEN
HIRCC	—	—	—	—	—	—	HIRCF	HIRCEN
HXTC	—	—	—	—	—	HXTM	HXTF	HXTEN
LXTC	—	—	—	—	—	—	LXTF	LXTEN

System Operating Mode Control Registers List

SCC Register

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

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

Bit 4 Unimplemented, read as "0"

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

Bit 2 **FSS**: Low Frequency clock selection

0: LIRC
1: LXT

Bit 1 **FHIDEN**: High Frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing an "HALT" instruction.

Bit 0 **FSIDEN**: Low Frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing an "HALT" instruction. The low frequency oscillator is controlled by this bit together with the WDT function enable control. If this bit is cleared to 0 but the WDT function is enabled, the f_{WDT} clock will also be enabled.

HIRCC Register

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

Bit 7~2 Unimplemented, read as "0"

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

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

HXTC Register

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

Bit 7~3 Unimplemented, read as "0"

Bit 2 **HXTM**: HXT mode selection
0: HXT frequency ≤ 10 MHz
1: HXT frequency > 10 MHz
This bit is used to select the HXT oscillator operating mode. Note that this bit must be properly configured before the HXT is enabled. When the HXTEN bit is set to 1 to enable the HXT oscillator, it is invalid to change the value of this bit.

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

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

LXTC Register

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

Bit 7~2 Unimplemented, read as "0"

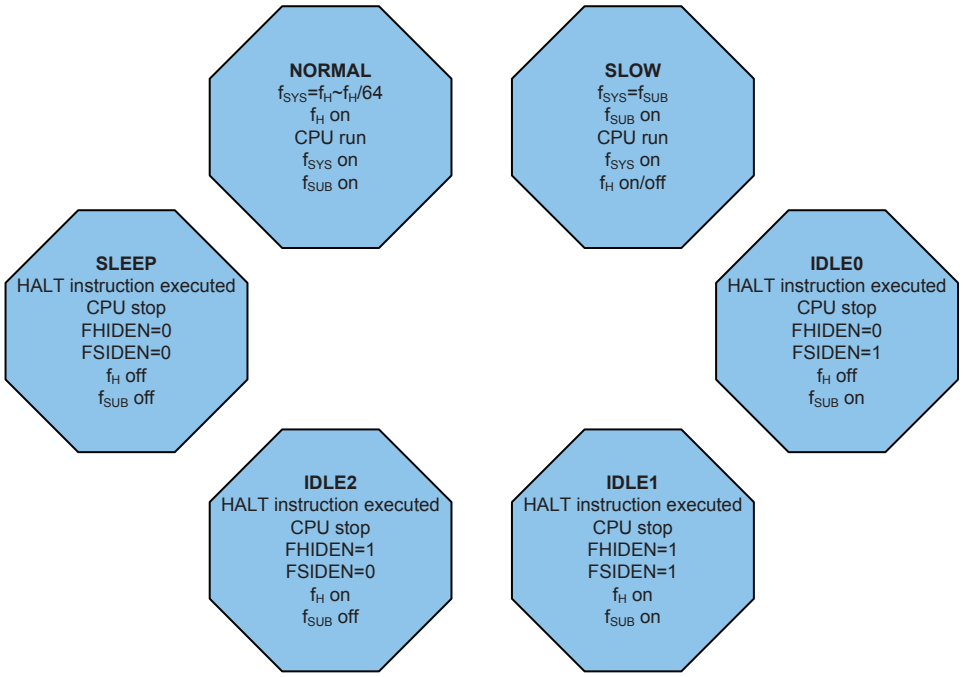
Bit 1 **LXTF**: LXT oscillator stable flag
0: LXT unstable
1: LXT stable
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set to 1 to enable the LXT oscillator, the LXTF bit will first be cleared to 0 and then set to 1 after the LXT oscillator is stable.

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

Operating Mode Switching

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

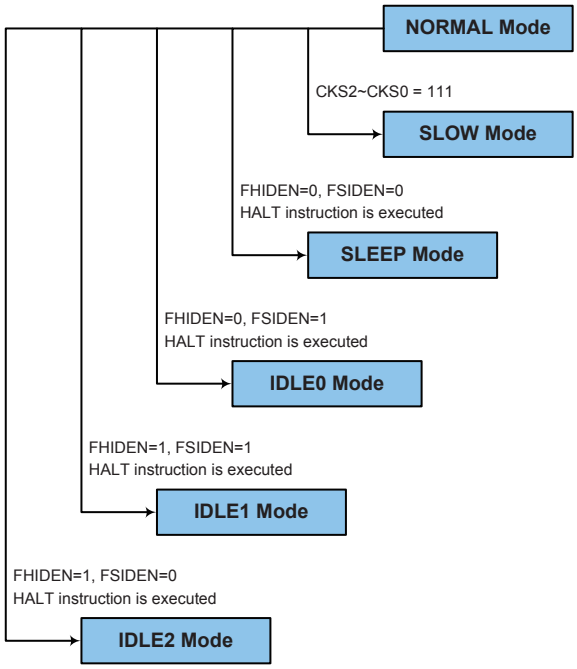
In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When = instruction is executed, whether the device enters the IDLE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



NORMAL Mode to SLOW Mode Switching

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

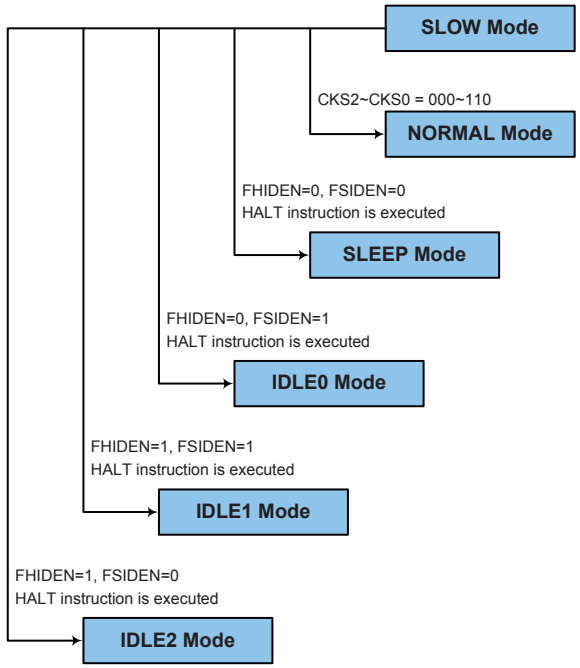
The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to NORMAL Mode Switching

In SLOW mode the system clock is derived from f_{SUB}. When system clock is switched back to the NORMAL 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 NORMAL mode from the SLOW Mode. This is monitored using the HXTF bit in the HXTC register or the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the A.C. characteristics.



Entering the SLEEP Mode

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

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

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

In the IDLE1 and IDLE 2 Mode the high speed oscillator is on, if the peripheral function clock source is derived from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

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

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

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

If the system is woken up by an external or USB reset, these devices will experience a full system reset, however, if these devices are woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flags. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the "HALT" instruction. 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, the 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 wukk rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{WDT}, which can be sourced form either the LXT or LIRC oscillator determined by the FSS bit in the SCC register. The LIRC internal oscillator has an approximate frequency of 32 kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The LXT oscillator is supplied by an external 32.768kHz crystal. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. This register controls the overall operation of the Watchdog Timer.

WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT function enable control
10101: Disable
01010: Enabled
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after 2~3 f_{LIRC} clock cycles and the WRF bit in the RSTFC register will be set to 1.

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

RSTFC Register

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

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

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

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

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

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control and reset control of the Watchdog Timer. If the WE4~WE0 bits are set to a value of 10101B, the WDT function will be disabled. The WDT function will be enabled when the WE4~WE0 bits are set to a value of 01010B. If the WE4~WE0 bits are set to any other values other than 01010B and 10101B, it will reset the device after 2~3 f_{LIRC} clock cycles. After power on these bits will have a value of 01010B.

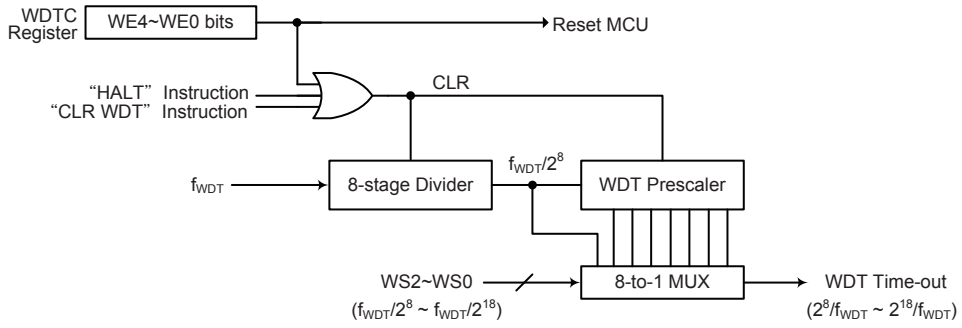
WE4 ~ WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

Watchdog Timer Enable/Disable Control

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

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

The maximum time out period is when the 2^{18} division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 second for the 2^{18} division ratio and a minimum timeout of 7.8ms for the 2^8 division ration.



Watchdog timer

Reset and Initialisation

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

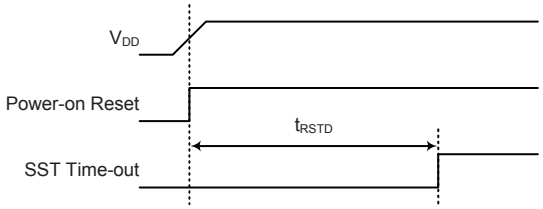
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All 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 both internally and externally.

Power-on Reset

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



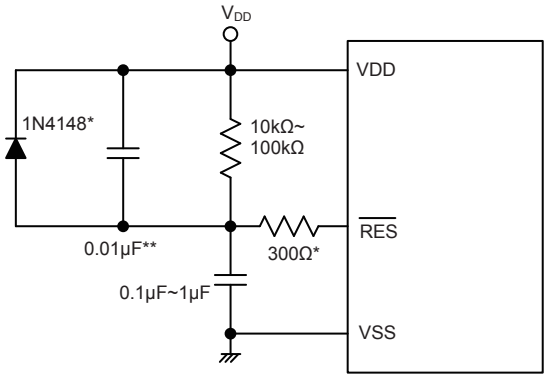
Note: t_{RSTD} is power-on delay with typical time = 50 ms

Power-On Reset Timing Chart

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

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

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

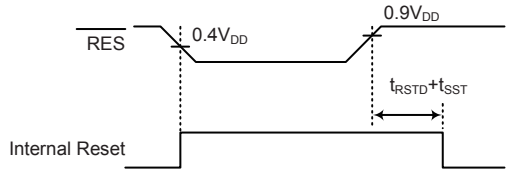


Note: "*" It is recommended that this component is added for added ESD protection.
"**" It is recommended that this component is added in environments where power line noise is significant.

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

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

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



Note: t_{RSTD} is power-on delay with typical time = 16.7 ms

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

There is an internal reset control register, RSTC, which is used to provide a reset when the device operates abnormally due to the environmental noise interference. If the content of the RSTC register is set to any value other than 01010101B or 10101010B, it will reset the device after 2~3 f_{LIRC} clock cycles. After power on the register will have a value of 01010101B.

RSTC7 ~ RSTC0 Bits	Reset Function
01010101B	I/O pin function
10101010B	$\overline{\text{RES}}$ pin function
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: I/O pin function
10101010: $\overline{\text{RES}}$ pin function
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after 2~3 LIRC clock cycles and the RSTF bit in the RSTFC register will be set to 1.

- **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

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

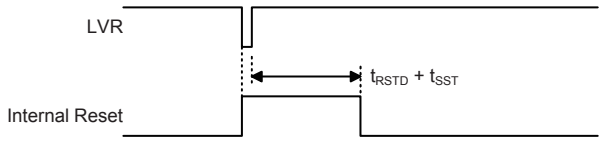
Bit 2 **LVRF**: LVR function reset flag
Described elsewhere.

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

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

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage, V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVD/LVR characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits have any other value, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the device after 2~3 f_{LIRC} clock cycles. When this happens, the LRF bit in the RSTFC register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the power down mode.



Note: t_{RSTD} is power-on delay with typical time = 50 ms

Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Other values: Generates a MCU reset – register is reset to POR value

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

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

- **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

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

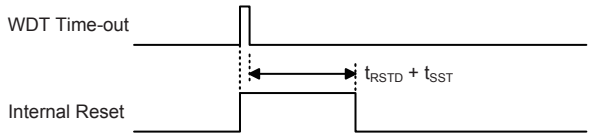
Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set to 1 when a specific low voltage reset condition occurs. Note that this bit can only be cleared to 0 by the application program.

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

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

Watchdog Time-out Reset during Normal Operation

The Watchdog time-out Reset during normal operation is the same as the hardware Low Voltage Reset except that the Watchdog time-out flag TO will be set to "1".

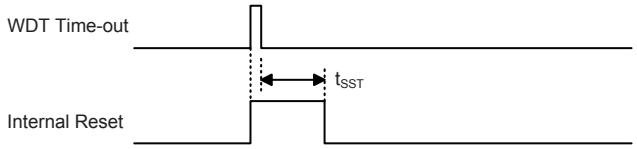


Note: t_{RSTD} is power-on delay with typical time = 16.7 ms

WDT Time-out Reset during NORMAL Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



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

Reset Initial Conditions

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

TO	PDF	Reset Function
0	0	Power-on reset
u	u	$\overline{\text{RES}}$, LVR or USB reset during NORMAL or SLOW Mode operation
1	u	WDT time-out reset during NORMAL or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

"u" stands for unchanged

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

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

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

Register	HT66F4360	HT66F4370	HT66F4390	Reset (Power On)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)*	USB-reset (Normal)	USB-reset (HALT)
PC	●	●	●	0000H	0000H	0000H	000H	0000H	0000H	0000H
IAR0	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP0	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
IAR1	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP1L	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP1H	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
ACC	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	●			--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
TBHP		●		-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
TBHP			●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
STATUS	●	●	●	xx00 xxxx	uuuu uuuu	uu01 uuuu	xx1u uuuu	uu11 uuuu	uuuu uuuu	uuuu uuuu
PBP	●			---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
PBP		●		---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
PBP			●	---- -000	---- -000	---- -000	---- -000	---- -uuu	---- -000	---- -000
IAR2	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP2L	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP2H	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
INTC0	●	●	●	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000

Register	HT66F4360	HT66F4370	HT66F4390	Reset (Power On)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)*	USB-reset (Normal)	USB-reset (HALT)
INTC1	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
INTC2	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
INTC3	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MFI0	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MFI1	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MFI2	•	•		--00 --00	--00 --00	--00 --00	--00 --00	--uu --uu	--00 --00	--00 --00
MFI2			•	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu	-000 -000	-000 -000
MFI3	•	•	•	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu	-000 -000	-000 -000
INTEG	•	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
PA	•	•	•	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PAC	•	•	•	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PAPU	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PAWU	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PB	•	•	•	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PBC	•	•	•	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PBPU	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PC	•	•	•	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PCC	•	•	•	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PCPU	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PE	•	•	•	1--- 1101	1--- 1101	1--- 1101	1--- 1101	u--- uuuu	1--- 1101	1--- 1101
PEC	•	•	•	1--- 1111	1--- 1111	1--- 1111	1--- 1111	u--- uuuu	1--- 1111	1--- 1111
PEPU	•	•	•	0--- 00-0	0--- 00-0	0--- 00-0	0--- 00-0	u--- uu-u	0--- 00-0	0--- 00-0
SCC	•	•	•	000- 0000	000- 0000	000- 0000	000- 0000	uuu- uuuu	000- 0000	000- 0000
HIRCC	•	•	•	---- --01	---- --01	---- --01	---- --01	---- --uu	---- --01	---- --01
HXTC	•	•	•	---- -000	---- -000	---- -000	---- -000	---- -uuu	---- -000	---- -000
LXTC	•	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
LVDC	•	•	•	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
LVRC	•	•	•	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu	0101 0101	0101 0101
WDTC	•	•	•	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu	0101 0011	0101 0011
RSTC	•	•	•	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu	0101 0101	0101 0101
RSTFC	•	•	•	---- 0x00	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
PLLC	•	•	•	---- --0-	---- --0-	---- --0-	---- --0-	---- --u-	---- --0-	---- --0-
PSCR	•	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
TB0C	•	•	•	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu	0--- -000	0--- -000
TB1C	•	•	•	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu	0--- -000	0--- -000
USB_STAT	•	•	•	11xx 000-	uuxx uuu-	uuxx uuu-	uuxx uuu-	uuxx uuu-	uuxx uuu-	uuxx uuu-
UINT	•	•	•	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
USC	•	•	•	1000 xxxx	uuuu xuux	uuuu xuux	uuuu xuux	uuuu xuux	uuuu 0100	uuuu 0100
UESR	•	•	•	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
UCC	•	•	•	000x 0xxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuu0 u000	uuu0 u000
AWR	•	•	•	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
STLI	•	•	•	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
STLO	•	•	•	xxxx xxx-	uuuu uuu-	uuuu uuu-	uuuu uuu-	uuuu uuu-	0000 000-	0000 000-
SIES	•	•	•	xx-x xxxx	ux-x xuuu	ux-x xuuu	ux-x xuuu	ux-x xuuu	00-0 0000	00-0 0000
MISC	•	•	•	xxxx xxxx	xxuu uuxx	xxuu uuxx	xxuu uuxx	xxuu uuxx	0000 0000	0000 0000
UFIEN	•	•	•	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
UFOEN	•	•	•	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
UFC0	•	•	•	0000 00--	uuuu uu--	uuuu uu--	uuuu uu--	uuuu uu--	uuuu uu--	uuuu uu--
UFC1	•	•	•	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu

Register	HT66F4360	HT66F4370	HT66F4390	Reset (Power On)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)*	USB-reset (Normal)	USB-reset (HALT)
EEA			•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
EED			•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FARL	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
FD0L	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
FD1L	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
FD2L	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
FD3L	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
FIFO0	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
FIFO1	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
FIFO2	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
FIFO3	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
FIFO4	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
FIFO5	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
FIFO6	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
FIFO7	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
DC2DC	•	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
CCR	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CSR	•	•	•	1000 0000	1000 0000	1000 0000	1000 0000	uuuu uuuu	1000 0000	1000 0000
CCCR	•	•	•	00xx x0x0	00xx x0x0	00xx x0x0	00xx x0x0	uuxx xuxu	00xx x0x0	00xx x0x0
CETU1	•	•	•	0--- -001	0--- -001	0--- -001	0--- -001	u--- -uuu	0--- -001	0--- -001
CETU0	•	•	•	0111 0100	0111 0100	0111 0100	0111 0100	uuu uuuuu	0111 0100	0111 0100
CGT1	•	•	•	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
CGT0	•	•	•	0000 1100	0000 1100	0000 1100	0000 1100	uuuu uuuu	0000 1100	0000 1100
CWT2	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CWT1	•	•	•	0010 0101	0010 0101	0010 0101	0010 0101	uuuu uuuu	0010 0101	0010 0101
CWT0	•	•	•	1000 0000	1000 0000	1000 0000	1000 0000	uuuu uuuu	1000 0000	1000 0000
CIER	•	•	•	0-00 0000	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu	0-00 0000	0-00 0000
CIPR	•	•	•	0-00 0000	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu	0-00 0000	0-00 0000
CTXB	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CRXB	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STMC0	•	•	•	0000 0---	0000 0---	0000 0---	0000 0---	uuuu u---	0000 0---	0000 0---
STMC1	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STMDL	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STMDH	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STMAL	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STMAH	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STMRP	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PTMC0	•	•	•	0000 0---	0000 0---	0000 0---	0000 0---	uuuu u---	0000 0---	0000 0---
PTMC1	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PTMDL	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PTMDH	•	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
PTMAL	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PTMAH	•	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
PTMRPL	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PTMRPH	•	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
SPI0C0	•	•	•	111- --00	111- --00	111- --00	111- --00	uuu- --uu	111- --00	111- --00
SPI0C1	•	•	•	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
SPI0D	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
SPI1C0	•	•	•	111- --00	111- --00	111- --00	111- --00	uuu- --uu	111- --00	111- --00

Register	HT66F4360	HT66F4370	HT66F4390	Reset (Power On)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)*	USB-reset (Normal)	USB-reset (HALT)
SPI1C1	•	•	•	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
SPI1D	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
U0SR	•	•	•	0000 1011	0000 1011	0000 1011	0000 1011	uuuu uuuu	0000 1011	0000 1011
U0CR1	•	•	•	0000 00x0	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu	0000 00x0	0000 00x0
U0CR2	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TXR_RXR0	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
BRG0	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
U1SR	•	•	•	0000 1011	0000 1011	0000 1011	0000 1011	uuuu uuuu	0000 1011	0000 1011
U1CR1	•	•	•	0000 00x0	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu	0000 00x0	0000 00x0
U1CR2	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TXR_RXR1	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
BRG1	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
URFC	•	•	•	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu	-000 -000	-000 -000
ISOC	•	•	•	0000 -000	0000 -000	0000 -000	0000 -000	uuuu -uuu	0000 -000	0000 -000
SYSC	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PAS0	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PAS1	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PBS0	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PBS1	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PCS0	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PCS1	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PES0	•	•	•	0000 --00	0000 --00	0000 --00	0000 --00	uuuu –uu	0000 --00	0000 --00
PES1	•	•	•	00-- ----	00-- ----	00-- ----	00-- ----	uu-- ----	00-- ----	00-- ----
PF	•	•	•	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PFC	•	•	•	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PFPU	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PFS0	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PFS1	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
IICC0	•	•	•	---- 000-	---- 000-	---- 000-	---- 000-	---- uuu-	---- 000-	---- 000-
IICC1	•	•	•	1000 0001	1000 0001	1000 0001	1000 0001	uuuu uuuu	1000 0001	1000 0001
IICD	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
IICA	•	•	•	0000 000-	0000 000-	0000 000-	0000 000-	uuuu uuu-	0000 000-	0000 000-
IICTOC	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
SADOL (ADRFS=0)	•	•	•	xxxx ----	xxxx ----	xxxx ----	xxxx ----	uuuu ----	xxxx ----	xxxx ----
SADOL (ADRFS=1)	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
SADOH (ADRFS=0)	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
SADOH (ADRFS=1)	•	•	•	---- xxxx	---- xxxx	---- xxxx	---- xxxx	---- uuuu	---- xxxx	---- xxxx
SADC0	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
SADC1	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CMP0C	•	•	•	-000 00--	-000 00--	-000 00--	-000 00--	-uuu uu--	-000 00--	-000 00--
CMP0VOS	•	•	•	-001 0000	-001 0000	-001 0000	-001 0000	-uuu uuuu	-001 0000	-001 0000
CMP1C	•	•	•	-000 00--	-000 00--	-000 00--	-000 00--	-uuu uu--	-000 00--	-000 00--
CMP1VOS	•	•	•	-001 0000	-001 0000	-001 0000	-001 0000	-uuu uuuu	-001 0000	-001 0000
EEC			•	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu	---- 0000	---- 0000
FRCR	•	•	•	xx-0 ---0	xx-0 ---0	xx-0 ---0	xx-0 ---0	uu-u ---u	xx-0 ---0	xx-0 ---0
FCR	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000

Register	HT66F4360	HT66F4370	HT66F4390	Reset (Power On)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)*	USB-reset (Normal)	USB-reset (HALT)
FARH	●			--xx xxxx	--xx xxxx	--xx xxxx	--xx xxxx	--uu uuuu	--xx xxxx	--xx xxxx
FARH		●		-xxx xxxx	-xxx xxxx	-xxx xxxx	-xxx xxxx	-uuu uuuu	-xxx xxxx	-xxx xxxx
FARH			●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
FD0H	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
FD1H	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
FD2H	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
FD3H	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
CTM0C0	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CTM0C1	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CTM0DL	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CTM0DH	●	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
CTM0AL	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CTM0AH	●	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
CTM1C0	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CTM1C1	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CTM1DL	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CTM1DH	●	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
CTM1AL	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CTM1AH	●	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00

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

Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
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
PE	PE7	—	—	—	PE3	PE2	PE1	PE0
PEC	PEC7	—	—	—	PEC3	PEC2	—	PEC0
PEPU	PEPU7	—	—	—	PEPU3	PEPU2	—	PEPU0
PF	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0

I/O Registers List

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

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the relevant pull-high control registers 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 an input or NMOS output. Otherwise, the pull-high resistors can not be enabled.

PAPU Register

Bit	7	6	5	4	3	2	1	0
Name	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
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 **PAPU7~PAPU0**: Port A bit 7~bit 0 pull-high function control
0: Disable
1: Enable

Port A Wake-up

The HALT instruction forces the microcontroller into the SLEEP or IDLE Mode which preserves power, a feature that is important for battery and other low-power applications. Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the Port A pins from high to low. This function is especially suitable for applications that can be woken up via external switches. Each pin on Port A can be selected individually to have this wake-up feature using the PAWU register. Note that the wake-up function can be controlled by the wake-up control registers only when the pin-shared functional pin is selected as general purpose input/output and the MCU enters the Power down mode.

PAWU Register

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

Bit 7~0 **PAWU7~PAWU0**: Port A bit 7~bit 0 wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

Each Port has its own control register, known as PAC~PFC, which controls the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register.

However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PEC	PEC7	—	—	—	PEC3	PEC2	—	PEC0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0

PACn/PBCn/ PCCn/ PECn/ PFCn: I/O Port bit n Input/Output control
0: Output
1: Input

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. Each device includes Port "x" output function Selection register "n", labeled as PxSn, which can select the desired functions of the multi-function pin-shared pins.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. To select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. To correctly deselect the 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
PES0	PES07	PES06	PES05	PES04	—	—	PES01	PES00
PES1	D7	D6	—	—	—	—	—	—
PFS0	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
PFS1	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10

Pin-shared Function Selection Registers 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
00/01: PA3
10: TX0
11: C0+

Bit 5~4 **PAS05~PAS04**: PA2 pin-shared function selection
00/01/10/11: PA2

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00/01: PA1
10: RX0
11: C0X

Bit 1~0 **PAS01~PAS00**: PA0 pin-shared function selection
00/01/10/11: PA0

- **PAS1 Register**

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

Bit 7~6 **PAS17~PAS16**: PA7 pin-shared function selection
00/01: PA7
10: $\overline{\text{SCS0}}$
11: C1-

Bit 5~4 **PAS15~PAS14**: PA6 pin-shared function selection
00/01: PA6
10: SCK0
11: C1+

Bit 3~2 **PAS13~PAS12**: PA5 pin-shared function selection
00/01: PA5
10: SDI0
11: C1X

Bit 1~0 **PAS11~PAS10**: PA4 pin-shared function selection
00/01: PA4
10: SDO0
11: C0-

- **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
00/01: PB3
10: $\overline{SCS1}$
11: AN3

Bit 5~4 **PBS05~PBS04**: PB2 pin-shared function selection
00/01: PB2
10: SCK1
11: AN2

Bit 3~2 **PBS03~PBS02**: PB1 pin-shared function selection
00/01: PB1
10: SDI1
11: AN1

Bit 1~0 **PBS01~PBS00**: PB0 pin-shared function selection
00/01: PB0
10: SDO1
11: AN0

- **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
00/01: PB7
10: VREF
11: AN7

Bit 5~4 **PBS15~PBS14**: PB6 pin-shared function selection
00/01: PB6
10: RX1
11: AN6

Bit 3~2 **PBS13~PBS12**: PB5 pin-shared function selection
00/01: PB5
10: TX1
11: AN5

Bit 1~0 **PBS11~PBS10**: PB4 pin-shared function selection
00/01/10: PB4/PTPI/INT1
11: AN4

- **PCS0 Register**

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

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

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

Bit 3~2 **PCS03~PCS02**: PC1 pin-shared function selection
00/01/10: PC1
11: CTP1B

Bit 1~0 **PCS01~PCS00**: PC0 pin-shared function selection
00/01/10: PC0
11: CTP0B

- **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
00/01/10: PC7/STCK
11: SDA

Bit 5~4 **PCS15~PCS14**: PC6 pin-shared function selection
00/01/10: PC6/PTCK
11: SCL

Bit 3~2 **PCS13~PCS12**: PC5 pin-shared function selection
00/01/10/11: PC5/CTCK1

Bit 1~0 **PCS11~PCS10**: PC4 pin-shared function selection
00/01/10/11: PC4/CTCK0

- **PES0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PES07	PES06	PES05	PES04	—	—	PES01	PES00
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7~6 **PES07~PES06**: PE3 pin-shared function selection
00/01/10: PE3
11: OSC2

Bit 5~4 **PES05~PES04**: PE2 pin-shared function selection
00/01/10: PE2
11: OSC1

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **PES01~PES00**: PE0 pin-shared function selection
00/01/10: PE0
11: VDDIO

- **PES1 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	—	—	—	—	—	—
R/W	R/W	R/W	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

Bit 7~6 **D7~D6**: Reserved bits, can be read and written.

Bit 5~0 Unimplemented, read as "0"

- **PFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PFS07~PFS06**: PF3 pin-shared function selection
00/01/10: PF3
11: STP

Bit 5~4 **PFS05~PFS04**: PF2 pin-shared function selection
00/01/10: PF2
11: PTP

Bit 3~2 **PFS03~PFS02**: PF1 pin-shared function selection
00/01/10: PF1
11: CTP1

Bit 1~0 **PFS01~PFS00**: PF0 pin-shared function selection
00/01/10: PF0
11: CTP0

- **PFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10
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 **PFS17~PFS16**: PF7 pin-shared function selection
00/01/10/11: PF7

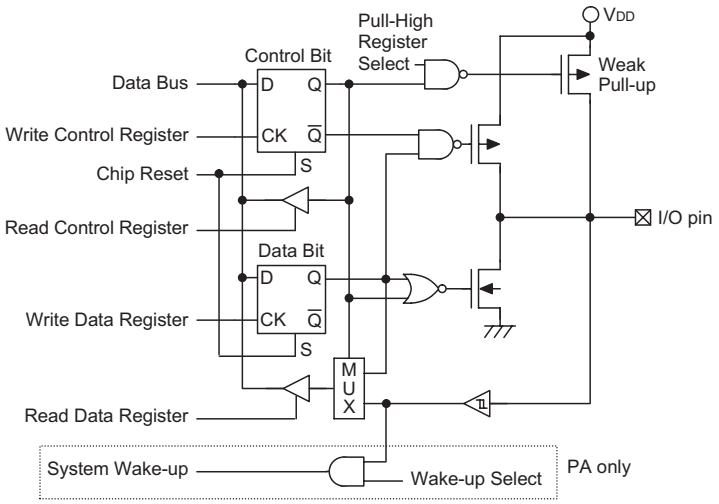
Bit 5~4 **PFS15~PFS14**: PF6 pin-shared function selection
00/01/10/11: PF6

Bit 3~2 **PFS13~PFS12**: PF5 pin-shared function selection
00/01/10/11: PF5

Bit 1~0 **PFS11~PFS10**: PF4 pin-shared function selection
00/01/10/11: PF4

I/O Pin Structures

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



Logic Function Input/Output Structure

Programming Considerations

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

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

Timer Modules – TM

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

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

Introduction

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

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

TM Function Summary

TM Operation

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

TM Clock Source

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

TM Interrupts

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

TM External Pins

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

The other xTM input pin, STPI or PTPI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the STIO1~STIO0 or PTIO1~PTIO0 bits in the STMC1 or PTMC1 register respectively. There is another capture input, PTCK, for PTM capture input mode, which can be used as the external trigger input source except the PTPI pin.

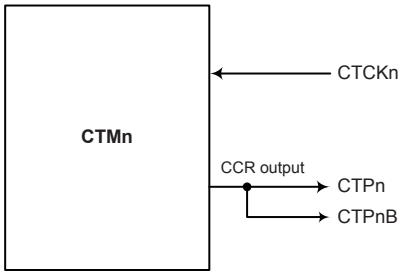
The TMs each have two output pins, xTPn and xTPnB. The xTPnB is the inverted signal of the xTPn output. The TM output pins can be selected using the corresponding pin-shared function selection bits described in the Pin-shared Function section. When the TM is in the Compare Match Output Mode, these pins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

Device	CTM		STM		PTM	
	Input	Output	Input	Output	Input	Output
HT66F4360 HT66F4370 HT66F4390	CTCK0 CTCK1	CTP0, CTP0B CTP1, CTP1B	STCK, STPI	STP, STPB	PTCK, PTPI	PTP, PTPB

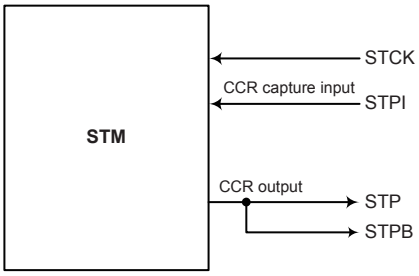
TM External Pins

TM Input/Output Pin Selection

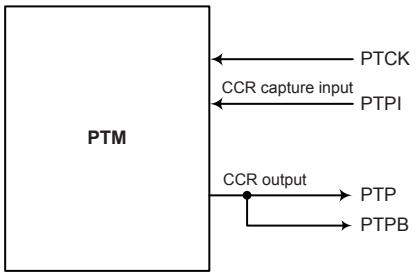
Selecting to have a TM input/output or whether to retain its other shared function is implemented using the relevant pin-shared function selection registers, with the corresponding selection bits in each pin-shared function register corresponding to a TM input/output pin. Configuring the selection bits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



CTM Function Pin Control Block Diagram – n = 0 or 1



STM Function Pin Control Block Diagram

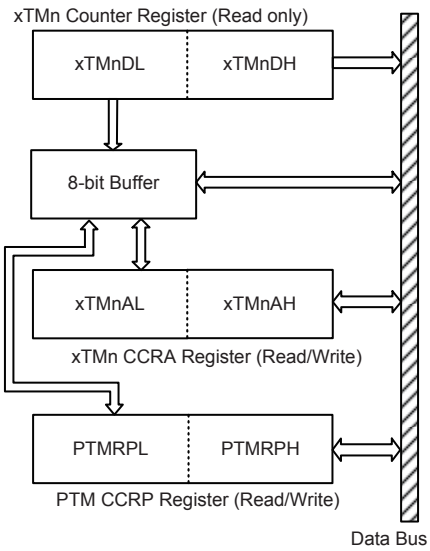


PTM Function Pin Control Block Diagram

Programming Considerations

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

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



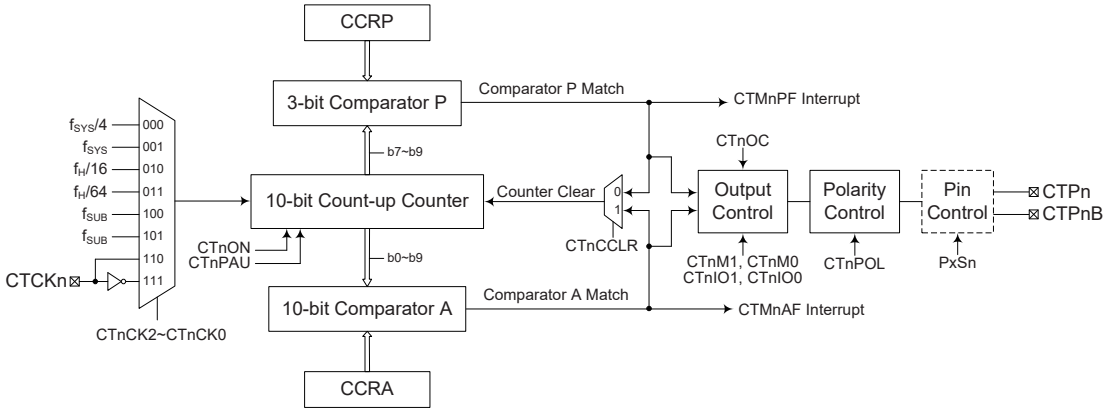
The following steps show the read and write procedures:

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

Compact Type TM – CTM

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

Device	CTM Core	CTM Input Pin	CTM Output Pin	Note
HT66F4360 HT66F4370 HT66F4390	10-bit CTM (CTM0, CTM1)	CTCK0, CTCK1	CTP0, CTP0B CTP1, CTP1B	n = 0 ~ 1



Compact Type TM Block Diagram – n = 0 or 1

Compact TM Operation

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

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

Compact Type TM Register Description

Overall operation of the Compact TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10-bit value, while a read/write register pair exists to store the internal 10-bit CCRA value. The remaining two registers are control registers which setup the different operating and control modes and 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 Registers List – n = 0 or 1

CTMnDL Register

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

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

CTMnDH Register

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

Bit 7~2 Unimplemented, read as "0"

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

CTMnAL Register

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

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

CTMnAH Register

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

Bit 7~2 Unimplemented, read as "0"

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

CTMnC0 Register

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

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

Bit 6~4 **CTnCK2~CTnCK0**: Select CTMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{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 while clearing the bit disables the CTMn. Clearing this bit to zero will stop the counter from counting and turn off the CTMn which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the CTMn is in the Compare Match Output Mode then the CTMn output pin will be reset to its initial condition, as specified by the CTnOC bit, when the CTnON bit changes from low to high.

Bit 2~0 **CTnRP2~CTnRP0**: CTMn CCRP 3-bit register, compared with the CTMn Counter bit 9 ~ bit 7
000: 1024 CTMn clocks
001: 128 CTMn clocks
010: 256 CTMn clocks
011: 384 CTMn clocks
100: 512 CTMn clocks
101: 640 CTMn clocks
110: 768 CTMn clocks
111: 896 CTMn clocks

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

CTMnC1 Register

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

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

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

Bit 5~4 **CTnIO1~CTnIO0**: Select CTMn external pin (CTPn) function
Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output
PWM Output Mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Undefined
Timer/Counter Mode
Unused

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

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

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

Bit 3 **CTnOC**: CTPn Output control

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

PWM Output Mode
0: Active low
1: Active high

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

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

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

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

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

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

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

Compact Type TM Operation Modes

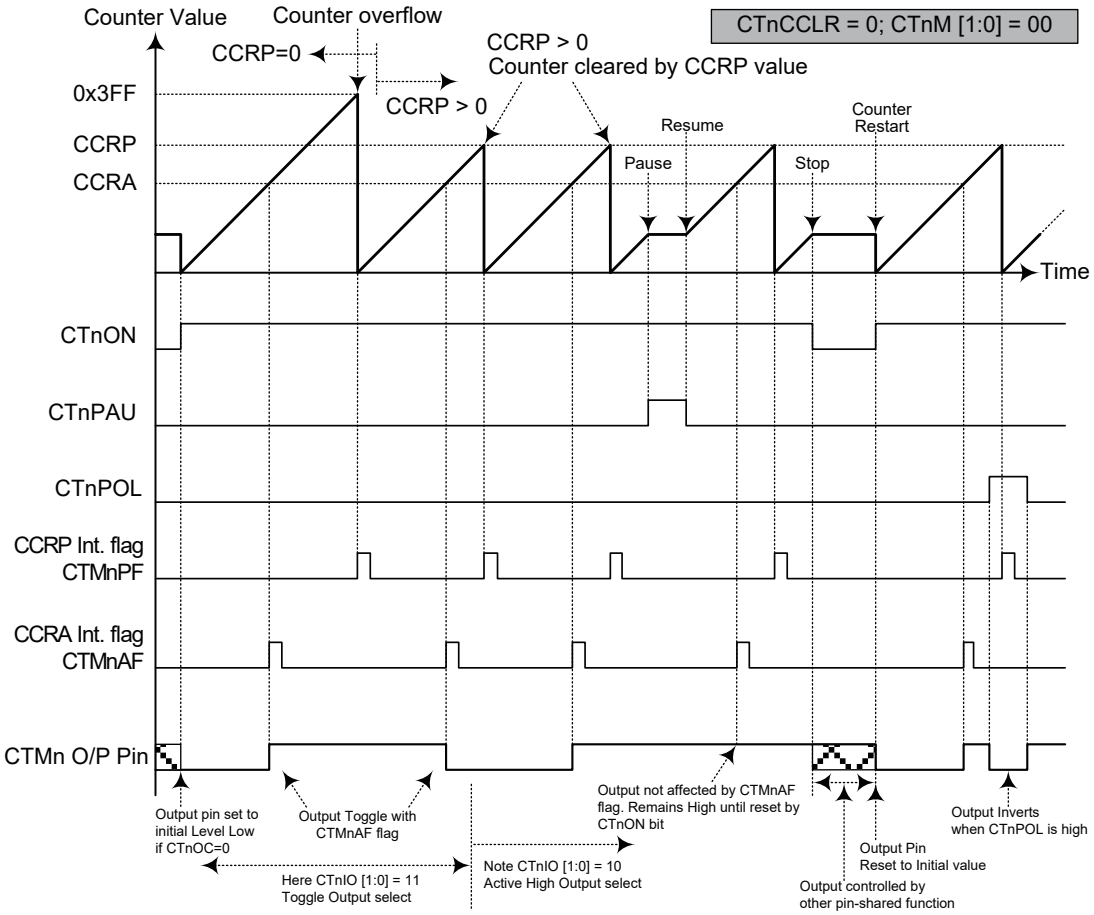
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

Compare Match Output Mode

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

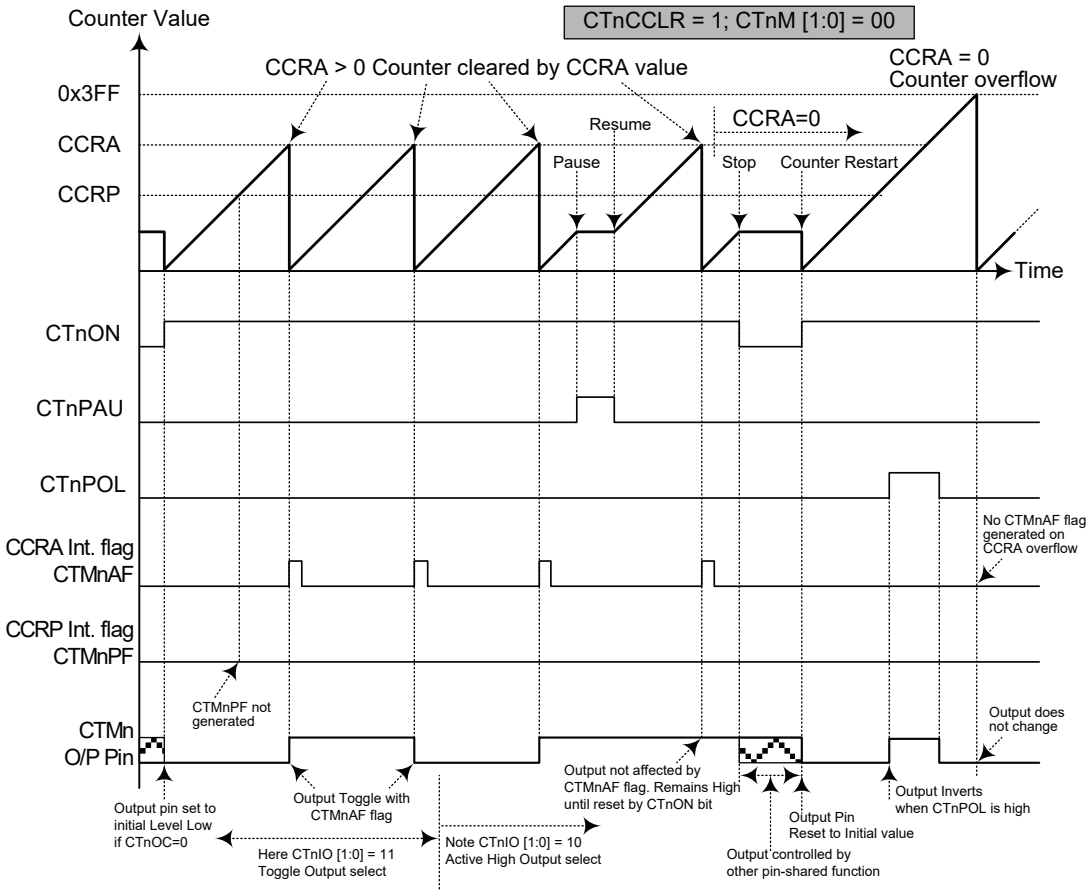
If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is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

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



Compare Match Output Mode – CTnCCLR = 0

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



Compare Match Output Mode – CTnCCLR = 1

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

Timer/Counter Mode

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

PWM Output Mode

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

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

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

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

CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	128	256	384	512	640	768	896	1024
Duty	CCRA							

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

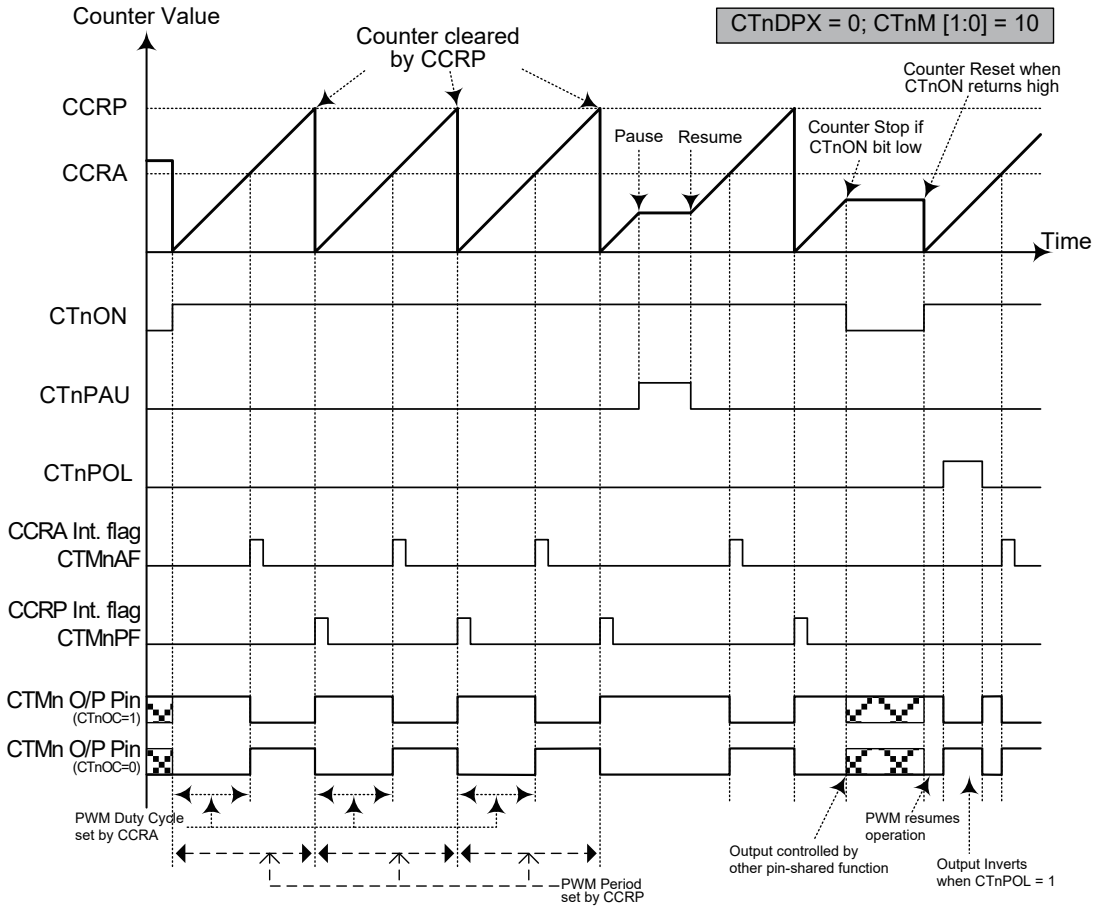
The CTMn PWM output frequency = ($f_{SYS}/4$) / 256 = f_{SYS} / 1024 = 15.625kHz, duty = 128 / 256= 50%.

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

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

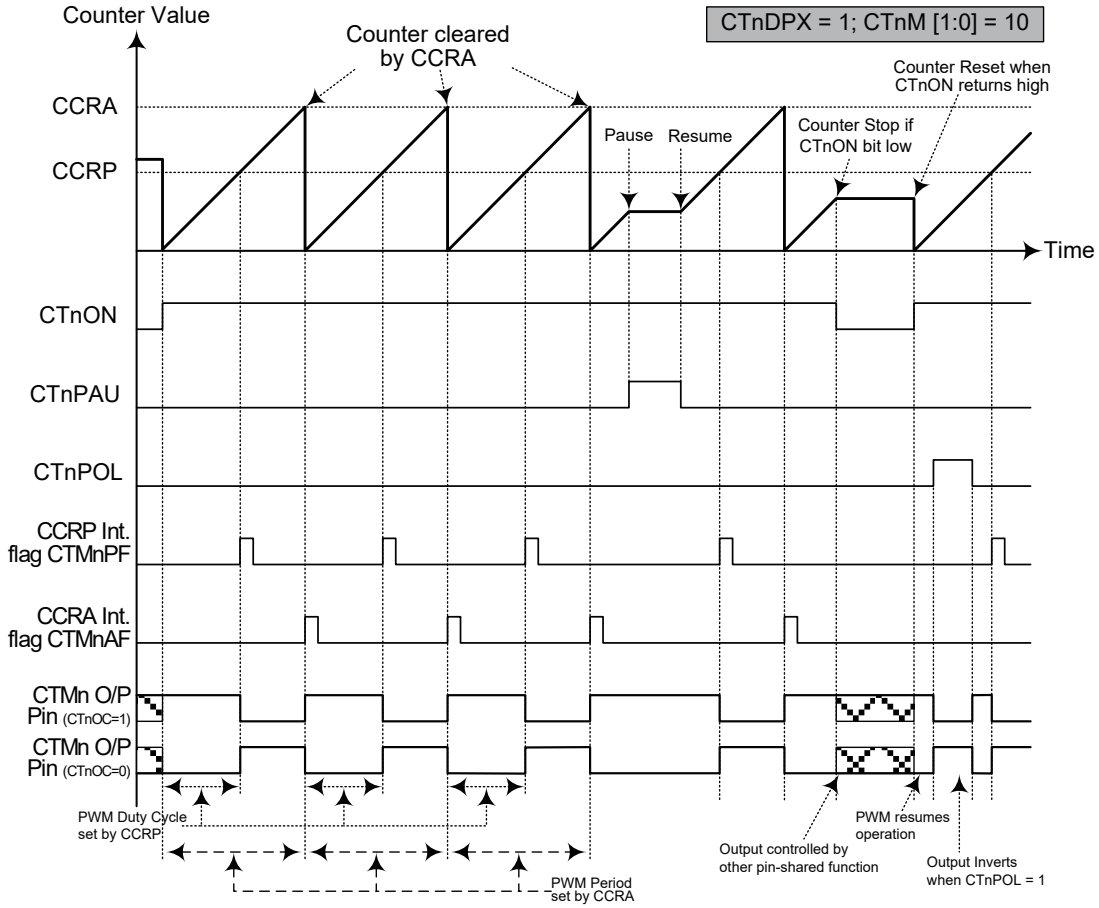
CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

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



PWM Output Mode – CTnDPX = 0

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



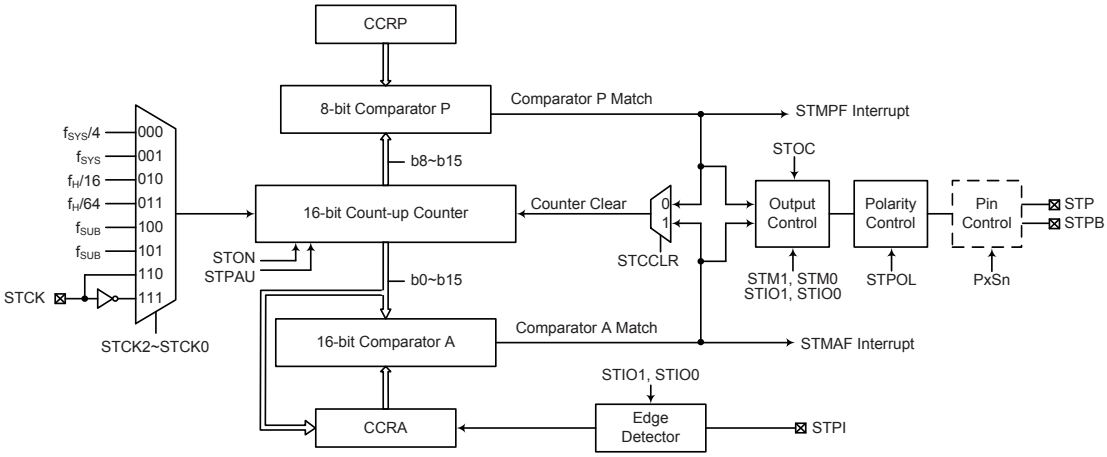
PWM Output Mode – CTnDPX = 1

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

Standard Type TM – STM

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

Device	STM Core	STM Input Pin	STM Output Pin
HT66F4360 HT66F4370 HT66F4390	16-bit STM	STCK, STPI	STP, STPB



Standard Type TM Block Diagram

Standard TM Operation

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

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

Standard Type TM Register Description

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

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

16-bit Standard TM Registers List

STMDL Register

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

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

STMDH Register

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

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

STMAL Register

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

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

STMAH Register

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

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

STMC0 Register

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

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

Bit 6~4 **STCK2~STCK0**: Select STM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: STCK rising edge clock
111: STCK falling edge clock
These three bits are used to select the clock source for the STM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **STON**: STM Counter On/Off control
0: Off
1: On
This bit controls the overall on/off function of the STM. Setting the bit high enables the counter to run while clearing the bit disables the STM. Clearing this bit to zero will stop the counter from counting and turn off the STM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the STM is in the Compare Match Output Mode then the STM output pin will be reset to its initial condition, as specified by the STOC bit, when the STON bit changes from low to high.

Bit 2~0 Unimplemented, read as “0”

STMC1 Register

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

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

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

Bit 5~4 **STIO1~STIO0**: Select STM external pin (STP or STPI) function

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

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

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

Timer/Counter Mode
Unused

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

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

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

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

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

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

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

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

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

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

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

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

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

STMRP Register

Bit	7	6	5	4	3	2	1	0
Name	STRP7	STRP6	STRP5	STRP4	STRP3	STRP2	STRP1	STRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **STRP7~STRP0**: STM CCRP 8-bit register, compared with the STM counter bit 15~bit 8

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

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

Standard Type TM Operation Modes

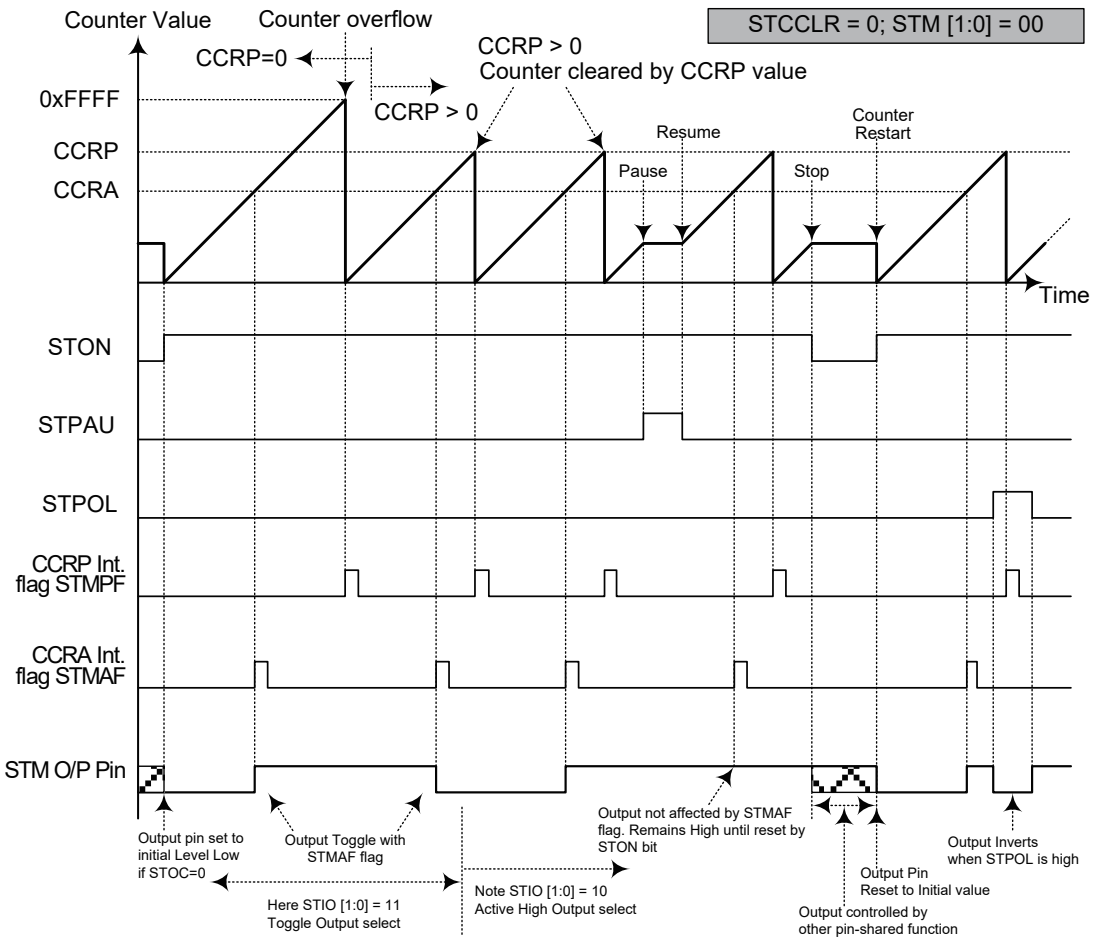
The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

Compare Match Output Mode

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

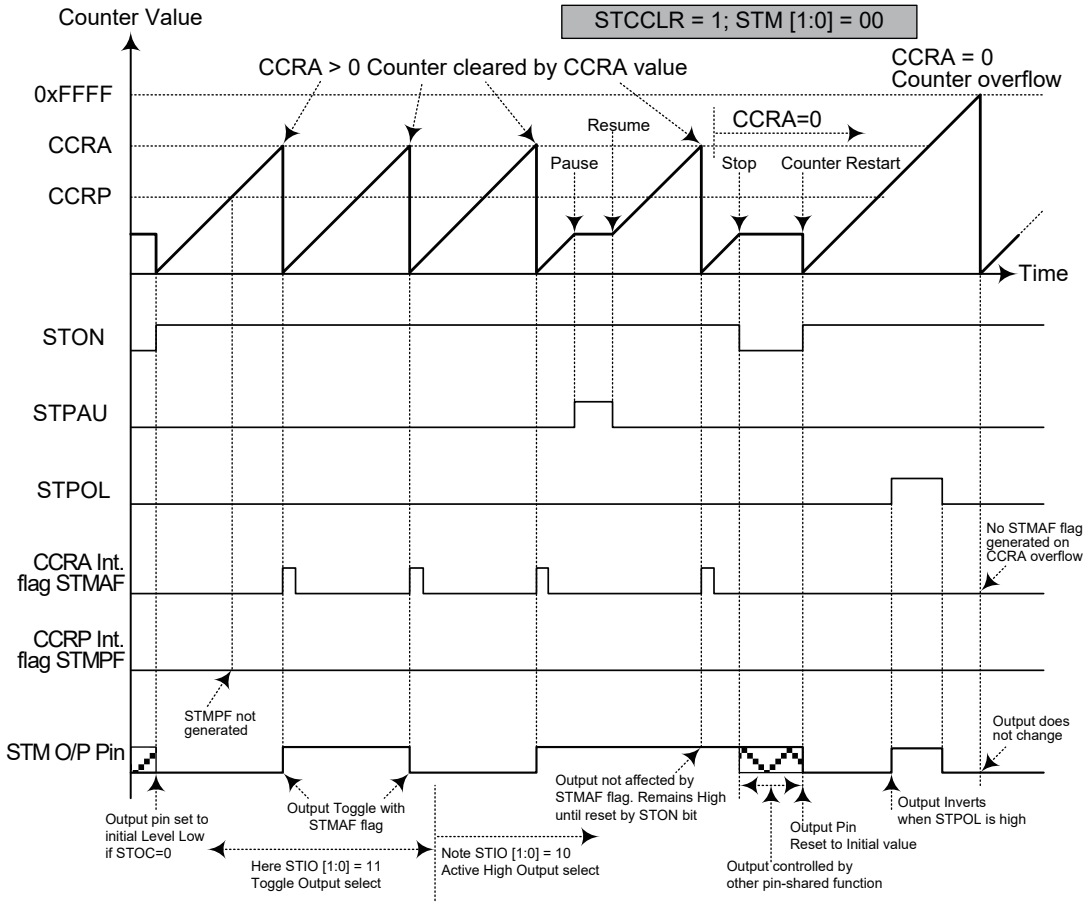
If the STCCLR bit in the STMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when STCCLR is high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

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



Compare Match Output Mode – STCCLR = 0

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



Compare Match Output Mode –STCCLR = 1

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

Timer/Counter Mode

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

PWM Output Mode

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

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

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

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

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

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

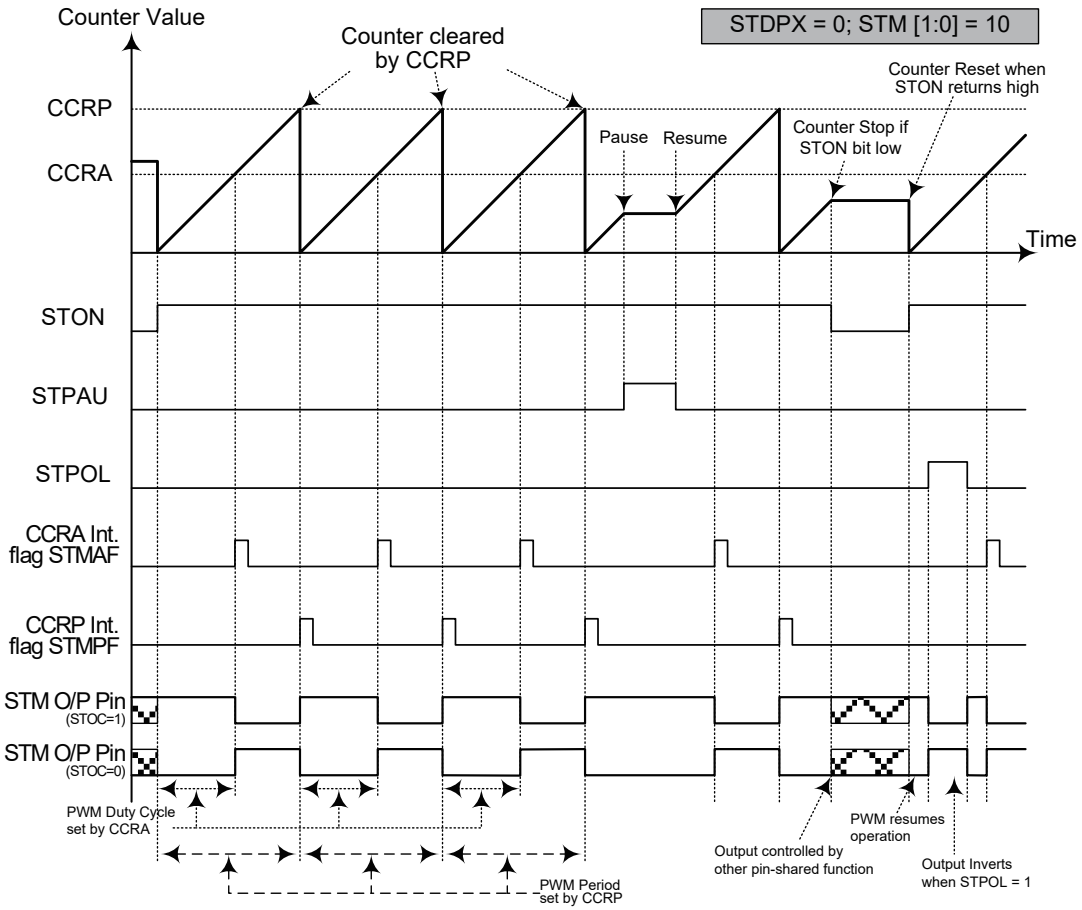
The STM PWM output frequency=($f_{SYS}/4$)/(2×256)=$f_{SYS}/2048$=7.8125kHz, duty=128/(2×256)= 25%.

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

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

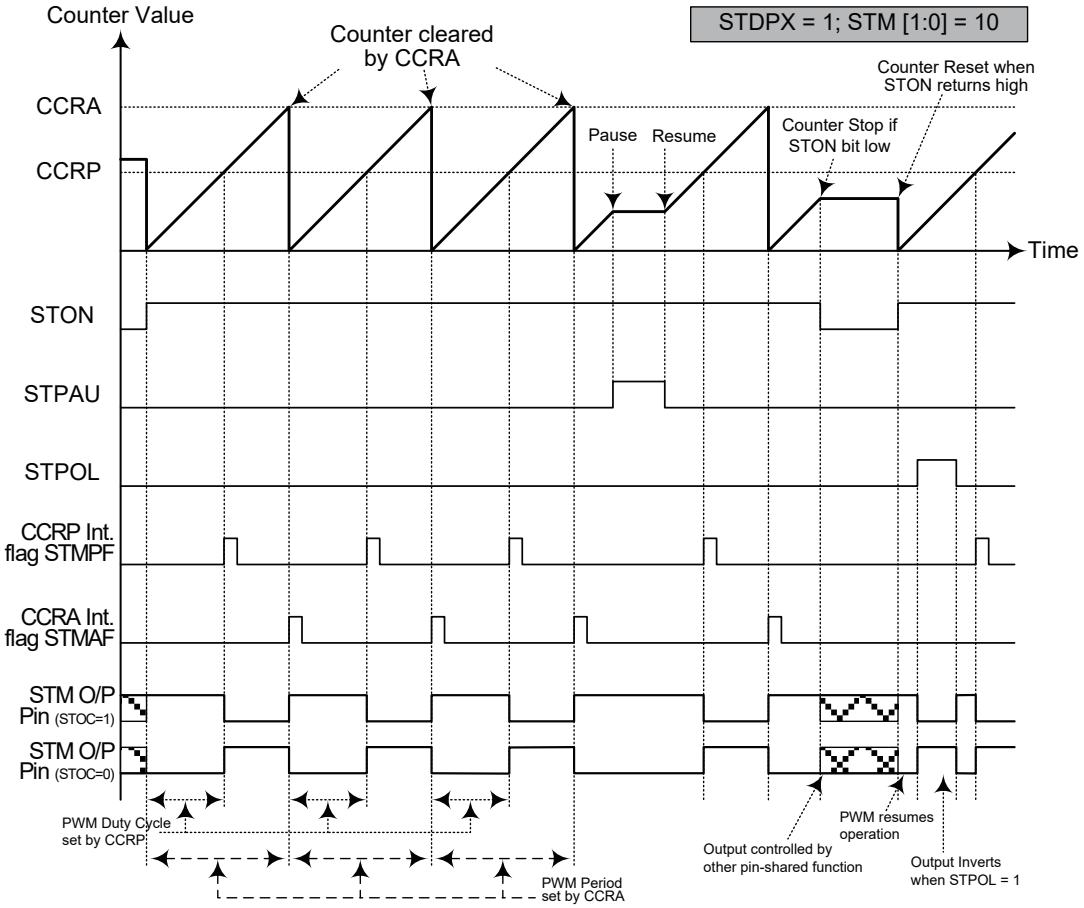
CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

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



PWM Output Mode – STDPX = 0

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



PWM Output Mode – STDPX = 1

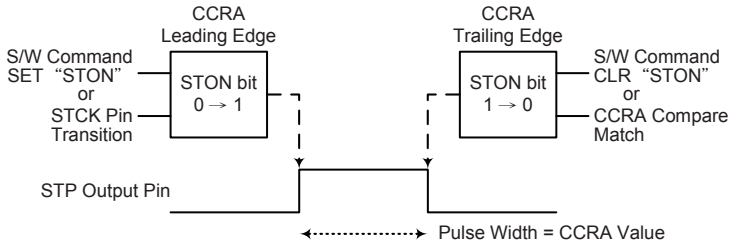
Note: 1. Here STDPX=1 – Counter cleared by CCRA
2. A counter clear sets the PWM Period
3. The internal PWM function continues even when STIO [1:0] = 00 or 01
4. The STCCLR bit has no influence on PWM operation

Single Pulse Output Mode

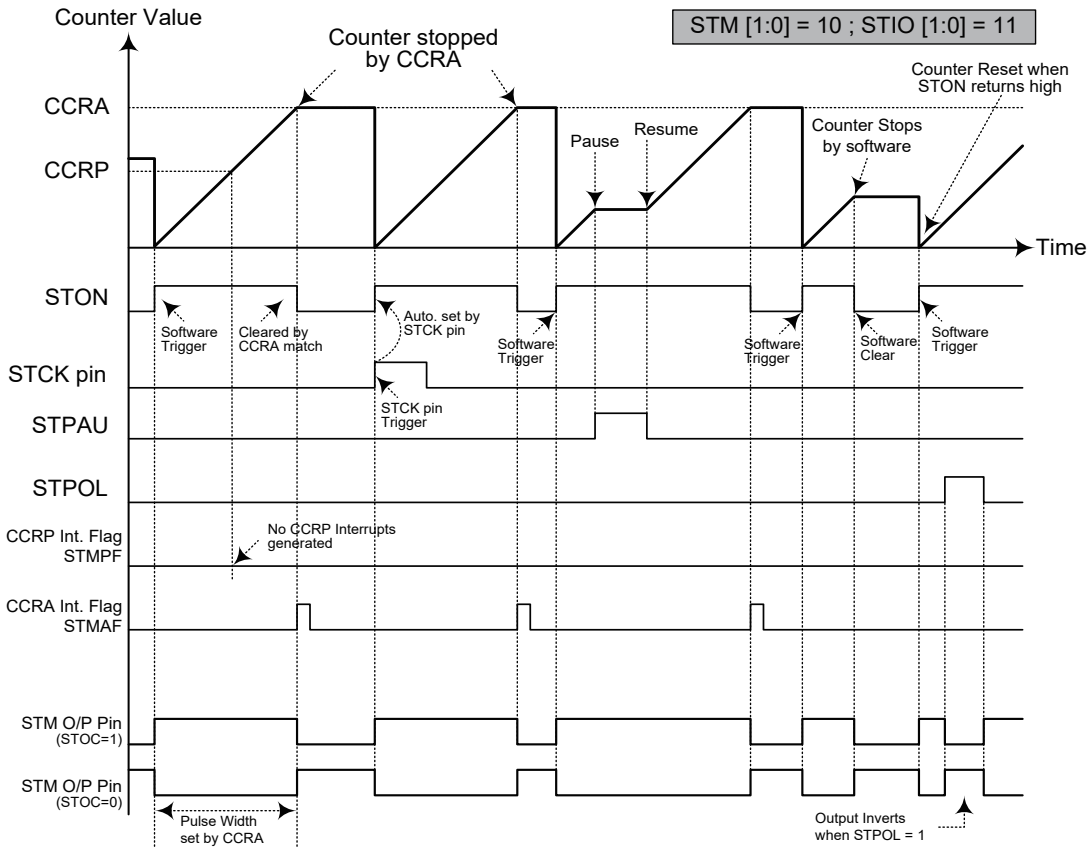
To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

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

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



Single Pulse Generation



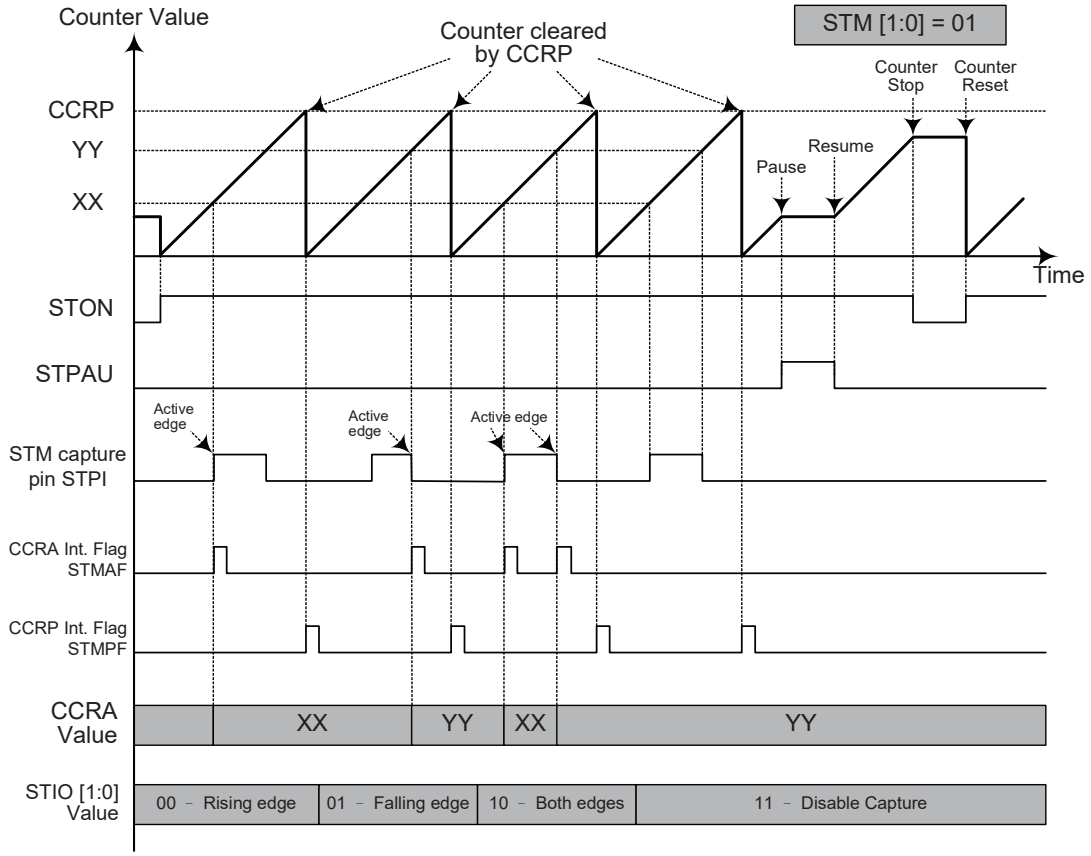
Single Pulse Mode

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

Capture Input Mode

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

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



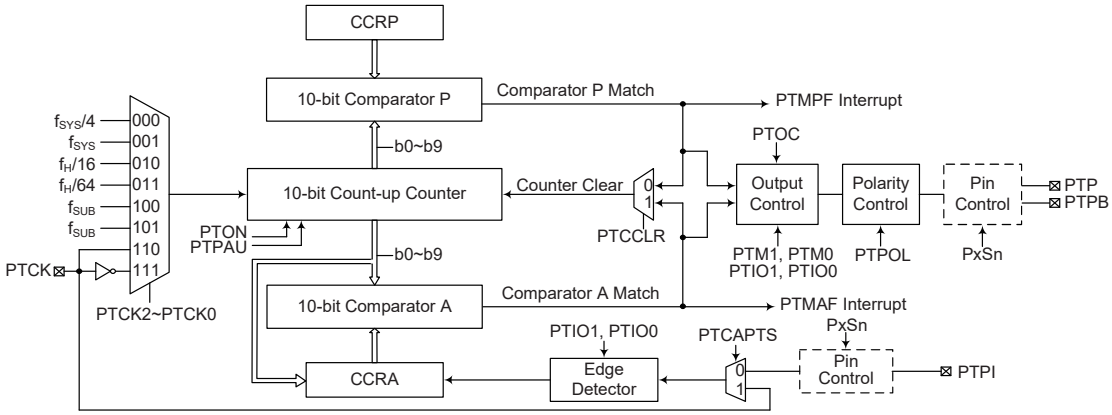
Capture Input Mode

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

Periodic Type TM – PTM

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

Device	PTM Core	PTM Input Pin	PTM Output Pin
HT66F4360 HT66F4370 HT66F4390	10-bit PTM	PTCK, PTPI	PTP, PTPB



Periodic Type TM Block Diagram

Periodic 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 respectively 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 PTON 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 PTM interrupt signal will also usually be generated. The Periodic Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control the output pins. All operating setup conditions are selected using relevant internal registers.

Periodic Type TM Register Description

Overall operation of the Periodic 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
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	—	—	—	—	—	—	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	—	—	—	—	—	—	D9	D8
PTMRPL	PTRP7	PTRP6	PTRP5	PTRP4	PTRP3	PTRP2	PTRP1	PTRP0
PTMRPH	—	—	—	—	—	—	PTRP9	PTRP8

Periodic TM Registers List

PTMDL 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 PTM Counter Low Byte Register bit 7 ~ bit 0
PTM 10-bit Counter bit 7 ~ bit 0

PTMDH 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 PTM Counter High Byte Register bit 1 ~ bit 0
PTM 10-bit Counter bit 9 ~ bit 8

PTMAL 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 PTM CCRA Low Byte Register bit 7 ~ bit 0
PTM 10-bit CCRA bit 7 ~ bit 0

PTMAH 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 PTM CCRA High Byte Register bit 1 ~ bit 0
PTM 10-bit CCRA bit 9 ~ bit 8

PTMRPL Register

Bit	7	6	5	4	3	2	1	0
Name	PTRP7	PTRP6	PTRP5	PTRP4	PTRP3	PTRP2	PTRP1	PTRP0
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 **PTRP7~PTRP0**: PTM CCRP Low Byte Register bit 7 ~ bit 0
PTM 10-bit CCRP bit 7 ~ bit 0

PTMRPH Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PTRP9	PTRP8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PTRP9~PTRP8**: PTM CCRP High Byte Register bit 1 ~ bit 0
PTM 10-bit CCRP bit 9 ~ bit 8

PTMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTPAU**: PTM 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 PTM 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 **PTCK2~PTCK0**: Select PTM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: PTCK rising edge clock
111: PTCK falling edge clock

These three bits are used to select the clock source for the PTM. 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 **PTON**: PTM Counter On/Off control
0: Off
1: On
This bit controls the overall on/off function of the PTM. Setting the bit high enables the counter to run while clearing the bit disables the PTM. Clearing this bit to zero will stop the counter from counting and turn off the PTM 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 PTM is in the Compare Match Output Mode then the PTM output pin will be reset to its initial condition, as specified by the PTOC bit, when the PTON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

PTMC1 Register

Bit	7	6	5	4	3	2	1	0
Name	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
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 **PTM1~PTM0**: Select PTM Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode
These bits setup the required operating mode for the PTM. To ensure reliable operation the PTM should be switched off before any changes are made to the PTM1 and PTM0 bits. In the Timer/Counter Mode, the PTM output pin control will be disabled.

Bit 5~4 **PTIO1~PTIO0**: Select PTM external pin PTP or PTPI function
Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output
PWM Output Mode/Single Pulse Output Mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Single Pulse Output
Capture Input Mode
00: Input capture at rising edge of PTPI or PTCK
01: Input capture at falling edge of PTPI or PTCK
10: Input capture at rising/falling edge of PTPI or PTCK
11: Input capture disabled
Timer/Counter Mode
Unused
These two bits are used to determine how the PTM output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the PTM is running.
In the Compare Match Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a compare match occurs from the Comparator A.

The PTM 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 PTM output pin should be setup using the PTOC bit in the PTMC1 register. Note that the output level requested by the PTIO1 and PTIO0 bits must be different from the initial value setup using the PTOC bit otherwise no change will occur on the PTM output pin when a compare match occurs. After the PTM output pin changes state, it can be reset to its initial level by changing the level of the PTON bit from low to high.

In the PWM Mode, the PTIO1 and PTIO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PTM output function is modified by changing these two bits. It is necessary to only change the values of the PTIO1 and PTIO0 bits only after the PTM has been switched off. Unpredictable PWM outputs will occur if the PTIO1 and PTIO0 bits are changed when the PTM is running.

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

Bit 2 **PTPOL**: PTM PTP Output polarity control
0: Non-inverted
1: Inverted

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

Bit 1 **PTCAPTS**: PTM Capture Triiger Source selection
0: From PTPI pin
1: From PTCK pin

Bit 0 **PTCCLR**: PTM 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 PTCCLR 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 PTCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

Periodic Type TM Operation Modes

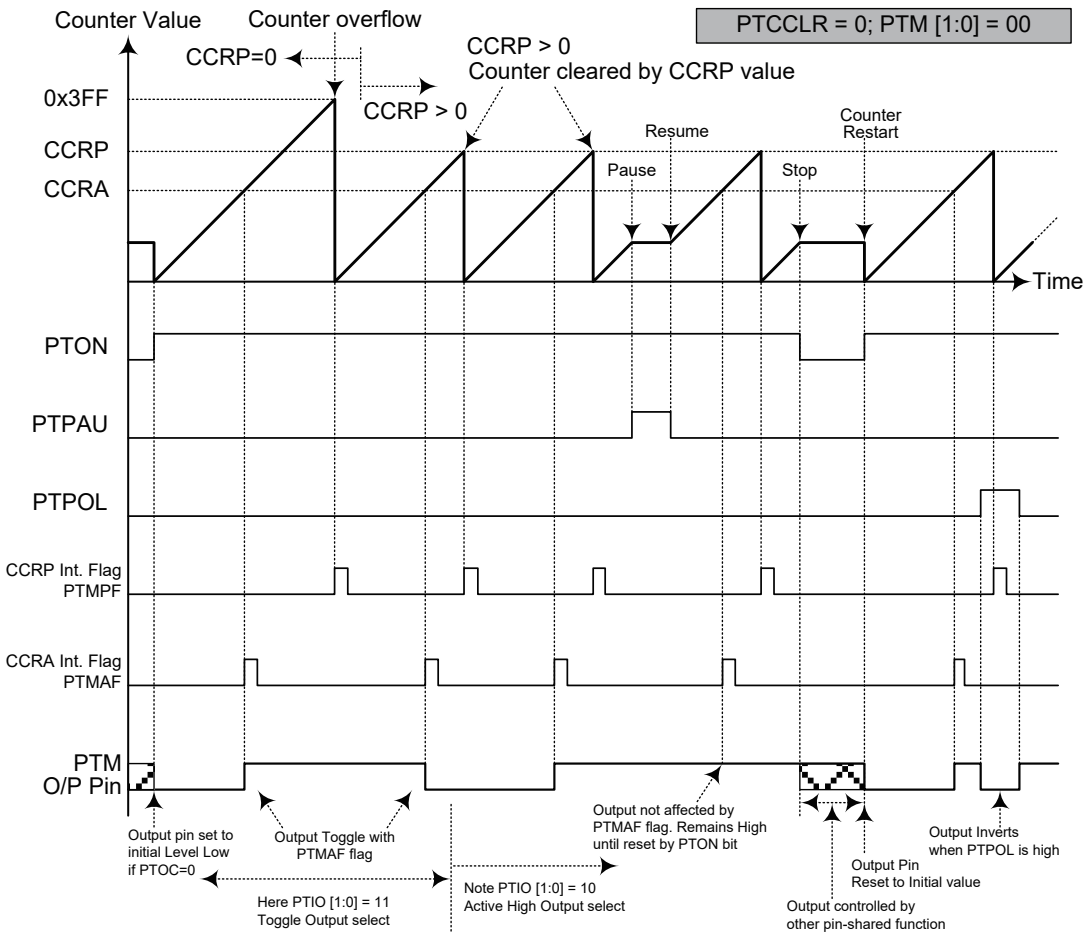
The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTM1 and PTM0 bits in the PTMC1 register.

Compare Match Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 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 PTCCLR 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 PTMAF and PTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

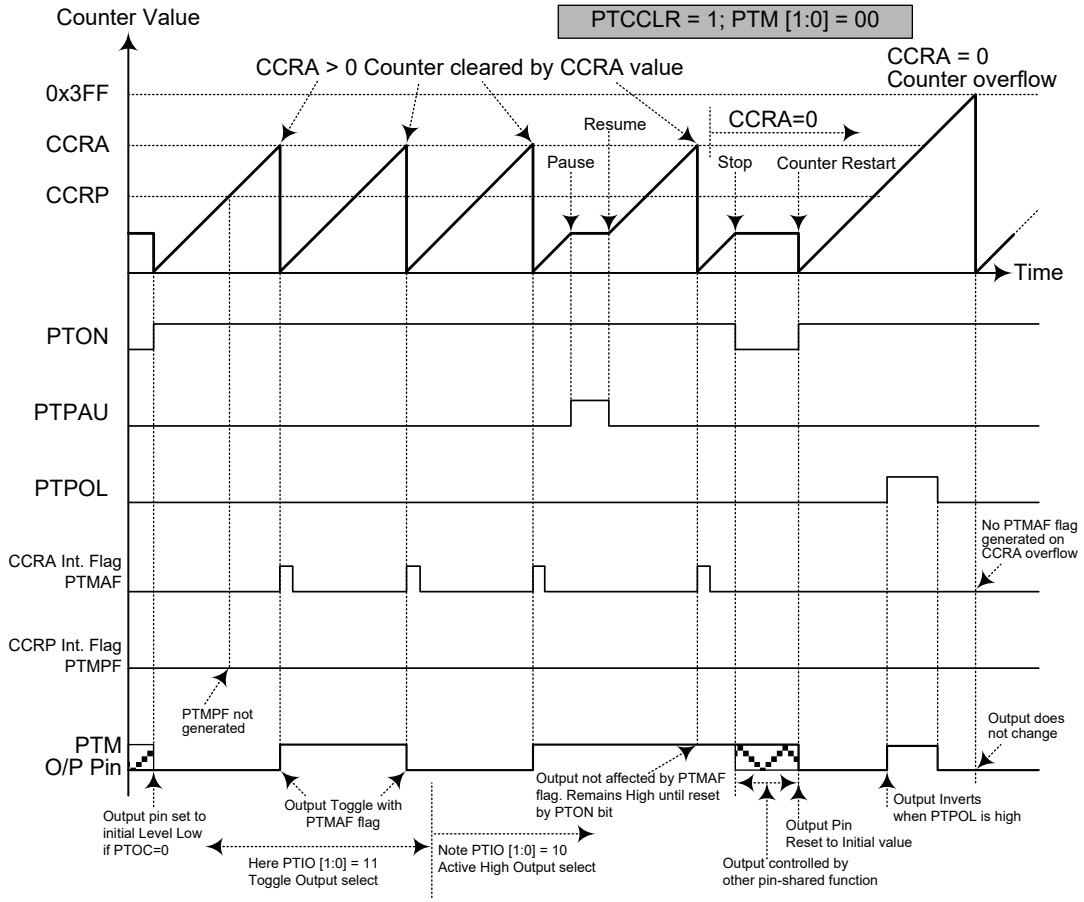
If the PTCCLR bit in the PTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTCCLR is high no PTMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”.

As the name of the mode suggests, after a comparison is made, the PTM output pin will change state. The PTM output pin condition however only changes state when a PTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the PTM output pin. The way in which the PTM output pin changes state are determined by the condition of the PTIO1 and PTIO0 bits in the PTMC1 register. The PTM output pin can be selected using the PTIO1 and PTIO0 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 PTM output pin, which is setup after the PTON bit changes from low to high, is setup using the PTOC bit. Note that if the PTIO1 and PTIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – PTCCLR = 0

Note: 1. With PTCCLR=0, a Comparator P match will clear the counter
2. The PTM output pin is controlled only by the PTMAF flag
3. The output pin is reset to its initial state by a PTON bit rising edge



Compare Match Output Mode – PTCCLR = 1

Note: 1. With PTCCLR=1, a Comparator A match will clear the counter
2. The PTM output pin is controlled only by the PTMAF flag
3. The output pin is reset to its initial state by a PTON bit rising edge
4. A PTMPF flag is not generated when PTCCLR =1

Timer/Counter Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 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 PTM 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 PTM 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 PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 10 respectively. The PWM function within the PTM 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 PTM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM mode, the PTCCLR 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 PTOC bit in the PTMC1 register is used to select the required polarity of the PWM waveform while the two PTIO1 and PTIO0 bits are used to enable the PWM output or to force the PTM output pin to a fixed high or low level. The PTPOL bit is used to reverse the polarity of the PWM output waveform.

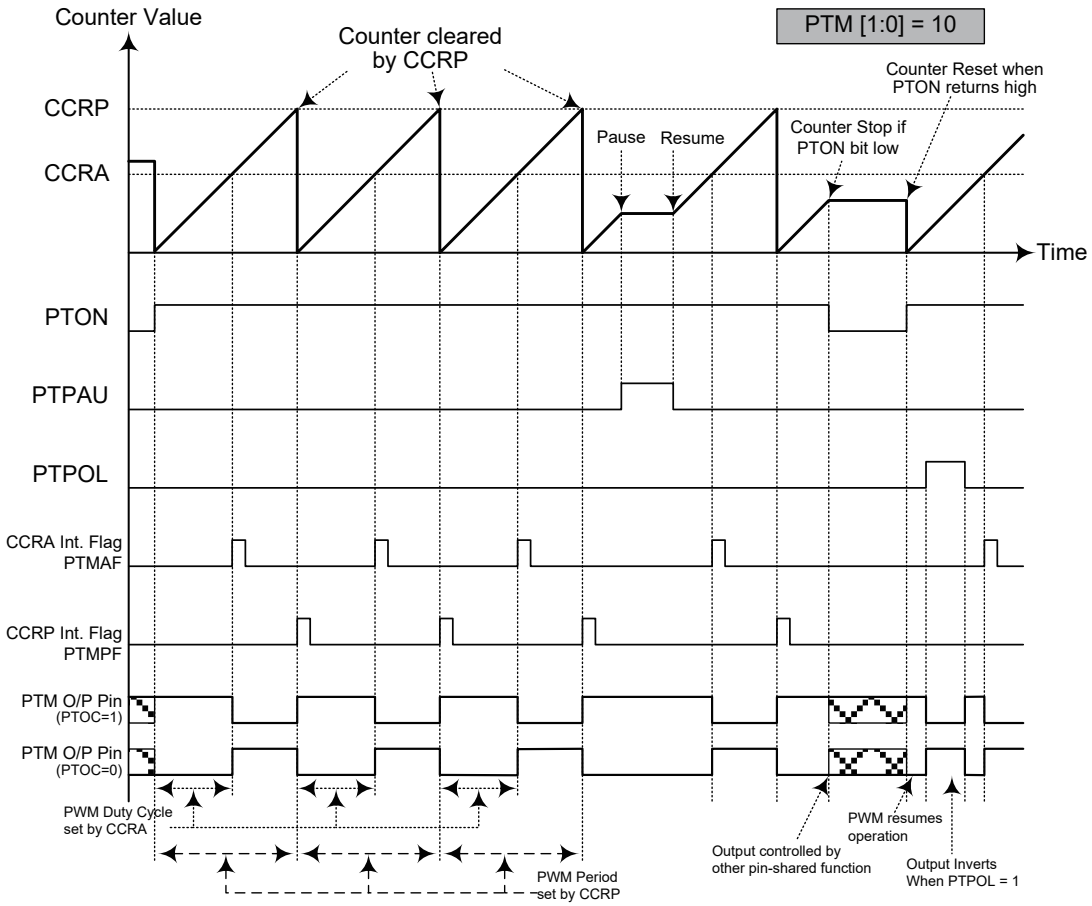
- **10-bit PTM, PWM Mode,**

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

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

The PTM 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

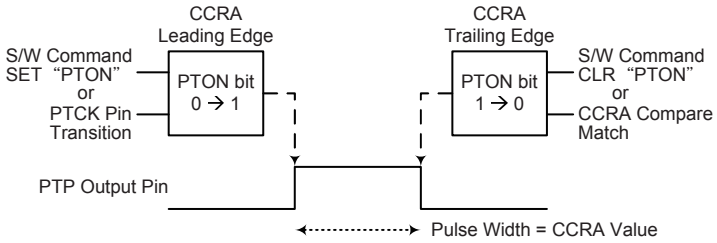
Note: 1. The counter is cleared by CCRP.
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTIO [1:0] = 00 or 01
4. The PTCCLR bit has no influence on PWM operation

Single Pulse Output Mode

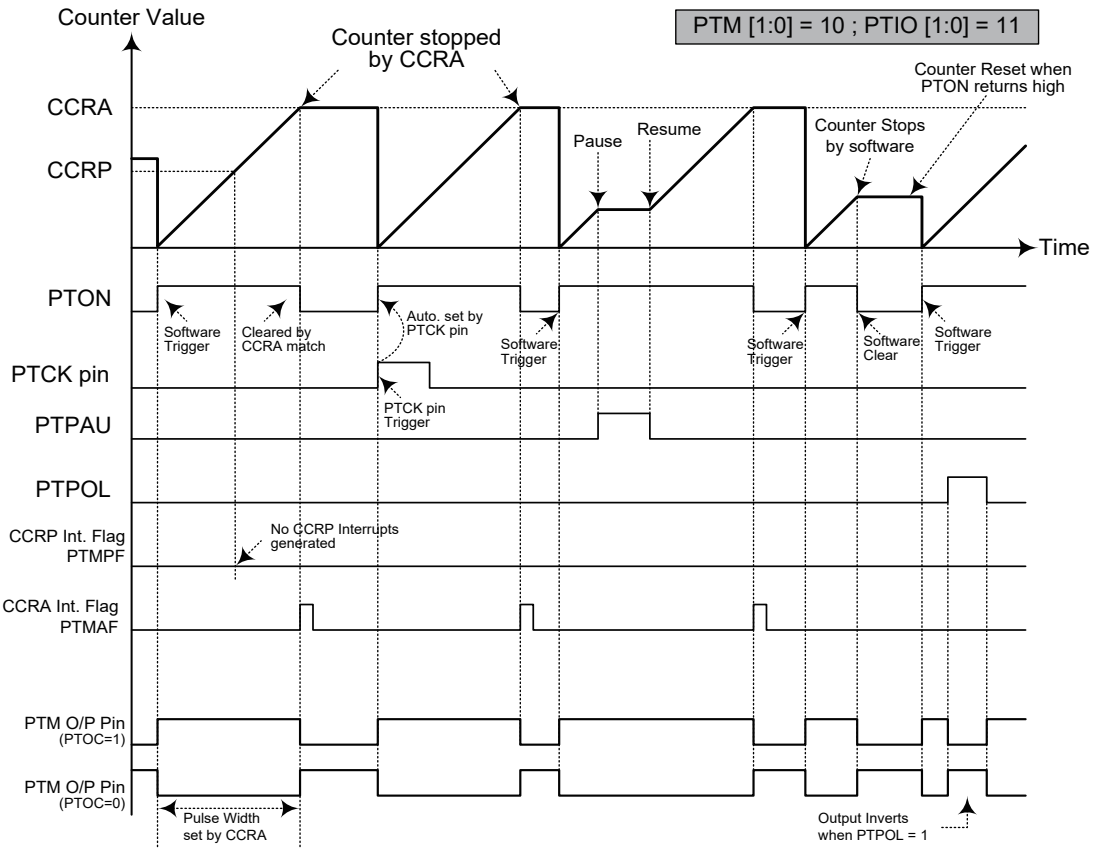
To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

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



Single Pulse Generation



Single Pulse Mode

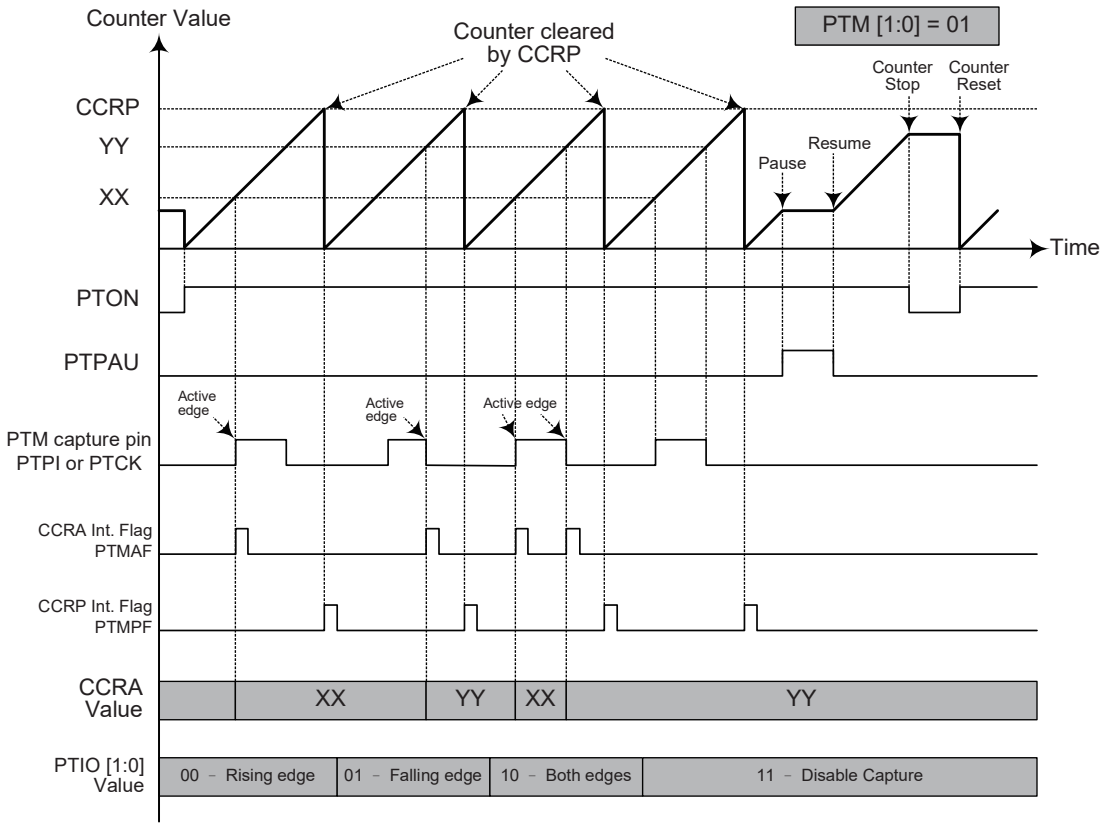
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse triggered by the PTCK pin or by setting the PTON bit high
4. A PTCK pin active edge will automatically set the PTON bit high.
5. In the Single Pulse Mode, PTIO [1:0] must be set to “11” and can not be changed.

Capture Input Mode

To select this mode bits PTM1 and PTM0 in the PTMC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPI or PTCK pin, selected by the PTCAPTS bit in the PTMC1 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTIO1 and PTIO0 bits in the PTMC1 register. The counter is started when the PTON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the PTPI or PTCK pin the present value in the counter will be latched into the CCRA registers and a PTM interrupt generated. Irrespective of what events occur on the PTPI or PTCK pin the counter will continue to free run until the PTON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTIO1 and PTIO0 bits can select the active trigger edge on the PTPI or PTCK pin to be a rising edge, falling edge or both edge types. If the PTIO1 and PTIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPI or PTCK pin, however it must be noted that the counter will continue to run.

As the PTPI or PTCK pin is pin shared with other functions, care must be taken if the PTM is in the Input Capture Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The PTCCLR, PTOC and PTPOL bits are not used in this Mode.



Capture Input Mode

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

Analog to Digital Converter

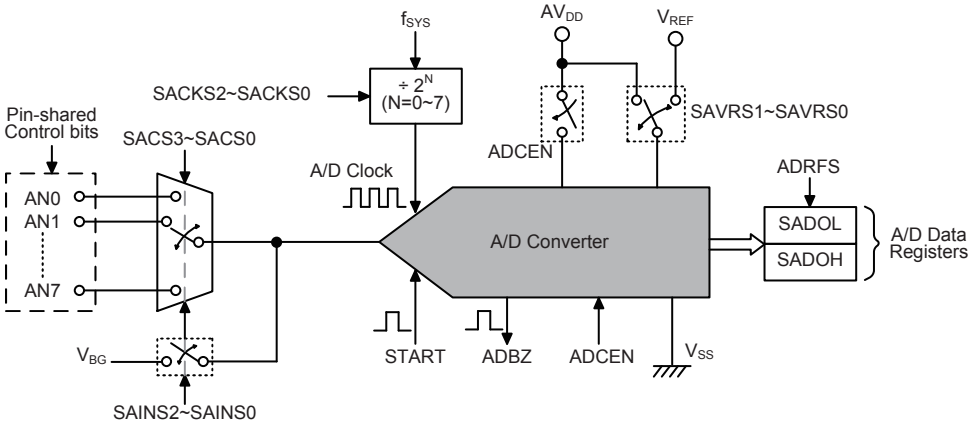
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the microcontroller, the need for external components is reduced significantly with the corresponding follow-on benefits of lower costs and reduced component space requirements.

A/D Overview

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

Device	External Input Channels	Internal Signal	A/D Channel Select Bits
HT66F4360 HT66F4370 HT66F4390	8: AN0~AN7	1: V_{BG}	SAINS2~SAINS0; SACS3~SACS0

The accompanying block diagram shows the internal structure of the A/D converter with temperature sensor together with its associated registers and control bits.



A/D Converter Structure

Registers Descriptions

Overall operation of the A/D converter is controlled using four registers. A read only register pair exists to store the A/D Converter data 12-bit value. Two registers, SADC0 and SADC1, are the control registers which setup the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

A/D Converter Registers List

A/D Converter Data Registers – SADOL, SADOH

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

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

A/D Converter Data Registers

A/D Converter Control Registers – SADC0, SADC1

To control the function and operation of the A/D converter, three control registers known as SADC0 and SADC1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external or internal analog signal inputs must be routed to the converter. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input. If the SAINS2~SAINS0 bits are set to “000”, the external analog channel input is selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected to be converted.

If the SAINS2~SAINS0 bits are set to “001”, the internal Bandgap reference voltage is selected to be converted. The internal Bandgap reference voltage comes from the LVD/LVR circuitry. More detail information is described in the LVD/LVR section. When the SAINS2~SAINS0 bits are set to “010”, “011” or “100”, there will be no internal analog signal to be routed to the converter.

When the SAINS2~SAINS0 bits are set to "101", "110" or "111", the external analog channel is selected to be connected to the A/D Converter as well as the "000" selection of the SAINS bit field.

Care must be taken when the internal analog signal is selected to be converted. If the internal analog signal is selected to be converted, the external analog channel input must never be selected as the A/D input by properly setting the SACS3~SACS0 bits. When the internal analog signal is selected, the SACS3~SACS0 bits should be configured with a value from 1000 to 1111. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

SAINS [2:0]	SACS [3:0]	Input Signals	Description
000, 101~111	0000~0111	AN0~AN7	External pin analog input
	1000~1111	—	Non-existed channel, input is floating.
001	1000~1111	V_{BG}	Internal Bandgap reference voltage
010, 011, 100	1000~1111	—	Reserved, connected to ground.

A/D Converter Input Signal Selection

The relevant pin-shared function selection bits determine which pins on I/O Ports are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

- **SADC0 Register**

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

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

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

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

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

Bit 3~0 **SACS3~SACS0**: A/D converter external analog channel input select
0000: External AN0 input
0001: External AN1 input
0010: External AN2 input
0011: External AN3 input
0100: External AN4 input
0101: External AN5 input
0110: External AN6 input
0111: External AN7 input
1xxx: Non-existed channel, the input will be floating if selected

- **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~5 **SAINS2~SAINS0**: A/D converter input signal select
000: External input – External analog channel input
001: Internal input – Internal Bandgap reference voltage, V_{BG}
010: Internal input – Reserved, connected to ground.
011: Internal input – Reserved, connected to ground.
100: Internal input – Reserved, connected to ground.
101~111: External input – External analog channel input

Care must be taken if the SAINS2~SAINS0 bits are set from "001" to "100" to select the internal analog signal to be converted. When the internal analog signal is selected to be converted, the external input pin must never be selected as the A/D input signal by properly setting the SACS3~SACS0 bits with a value from 1000 to 1111. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage select
00: VREF pin
01: Internal A/D converter power, AV_{DD}
1x: VREF pin

These bits are used to select the A/D converter reference voltage. Care must be taken if the SAVRS1~SAVRS0 bits are set to "01" to select the internal A/D converter power as the reference voltage source. When the internal A/D converter power is selected as the reference voltage, the VREF pin can not be configured as the reference voltage input by properly configuring the corresponding pin-shared function control bits. Otherwise, the external input voltage on VREF pin will be connected to the internal A/D converter power.

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source select
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

These bits are used to select the clock source for the A/D converter.

A/D Operation

The START bit in the SADC0 register is used to start the AD conversion. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated.

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

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the SACKS2~SACKS0 bits in the SADC1 register. Although the A/D clock source is determined by the system clock f_{SYS} and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK}, is from 0.5μs to 10μ, care must be taken for system clock frequencies. For example, as the system clock operates at a frequency of 12MHz, the SACKS2~SACKS0 bits should not be set to 000~010 or 111. Doing so will give A/D clock periods that are less than the minimum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

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

A/D Clock Period Examples

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

A/D Reference Voltage

The reference voltage supply to the A/D Converter can be supplied from the positive power supply pin, AV_{DD}, or from an external reference source supplied on pin VREF. The desired selection is made using the SAVRS1 and SAVRS0 bits. When the SAVRS bit field is set to "01", the A/D converter reference voltage will come from the AVDD pin. Otherwise, if the SAVRS bit field is set to any other value except "01", the A/D converter reference voltage will come from the VREF pin. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bits should be properly configured to disable other pin functions. However, if the internal A/D converter power is selected as the reference voltage, the VREF pin must not be configured as the reference voltage input function to avoid the internal connection between the VREF pin to A/D converter power AVDD.

A/D Converter Input Signals

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

There is an internal analog signal derived from the Bandgap reference voltage, which can be connected to the A/D converter as the analog input signal by configuring the SAINS2~SAINS0 bits. If the internal analog signal is selected to be converted, the external input channel determined by the SACS3~SACS0 bits must be switched to a non-existed A/D input channel by properly setting the SACS bit field with a value from 1000 to 1111.

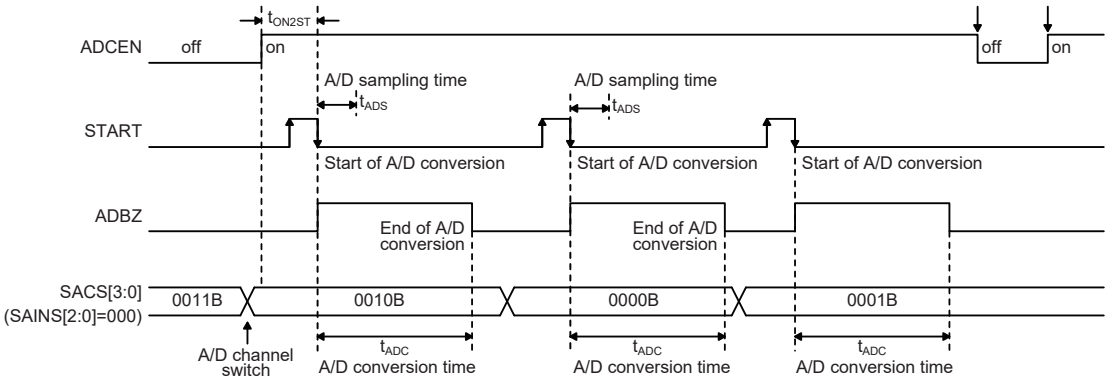
The A/D converter has its own reference voltage pin VREF however the reference voltage can also be supplied from the power supply pin, a choice which is made through the SAVRS1~SAVRS0 bits in the SADC1 register. The analog input values must not be allowed to exceed the value of the selected reference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

Conversion Rate and Timing Diagram

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

Maximum single A/D conversion rate = A/D clock period / 16

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



A/D Conversion Timing

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by properly programming the SACKS2~SACKS0 bits in the SADC1 register.
- Step 2
  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to one.
- Step 3
  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS2~SAINS0 bits
  Select the external channel input to be converted, go to Step 4.
  Select the internal analog signal to be converted, go to Step 5.
- Step 4
  If the A/D input signal comes from the external channel input selecting by configuring the SAINS bit field, the corresponding pins should be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS bit field. After this step, go to Step 6.
- Step 5
  Before the A/D input signal is selected to come from the internal analog signal by configuring the SAINS bit field, the corresponding external input pin must be switched to a non-existed channel input by setting the SACS3~SACS0 bits with a value from 1000 to 1111. The desired internal analog signal then can be selected by configuring the SAINS bit field. After this step, go to Step 6.
- Step 6
  Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register.
- Step 7
  Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.
- Step 8
  If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt bontrol bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.

- Step 9
  The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10
  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from ADRH and ADRL registers.

Note: When checking for the end of the conversion process, if the method of polling the ADBZ bit in the ADCR0 register is used, the interrupt enable step above can be omitted.

Programming Considerations

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

A/D Transfer Function

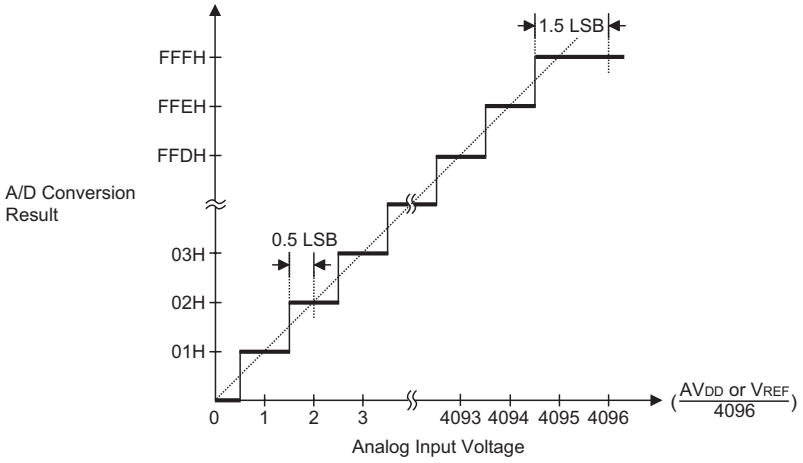
As the devices contain a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the reference voltage, AV_{DD} or V_{REF}, this gives a single bit analog input value of reference voltage value divided by 4096.

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

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

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

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



Ideal A/D Transfer Function

A/D Programming Examples

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

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

```
clr ADE               ; disable ADC interrupt
mov a,03H
mov SADC1,a           ; select fSYS/8 as A/D clock
set ADCEN
mov a,03h             ; setup PBS1 register to configure pin AN0
mov PBS0,a
mov a,20h
mov SADC0,a           ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr START             ; high pulse on start bit to initiate conversion
set START             ; reset A/D
clr START             ; start A/D
:
polling_EOC:
sz  ADBZ              ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC       ; continue polling
:
mov a,SADOL           ; read low byte conversion result value
mov ADRL_buffer,a     ; save result to user defined register
mov a,SADOH           ; read high byte conversion result value
mov ADRH_buffer,a     ; save result to user defined register
:
jmp start_conversion  ; start next A/D conversion
```

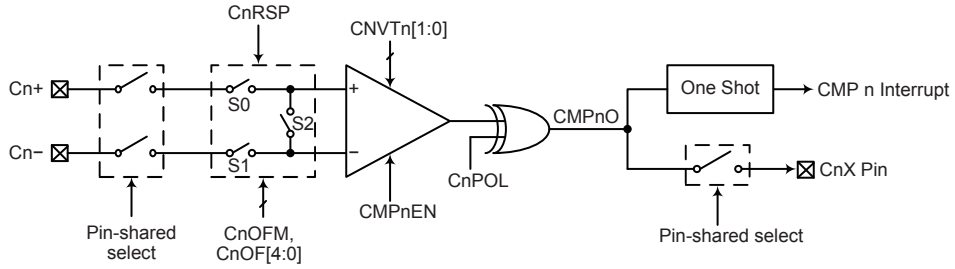
Example: using the interrupt method to detect the end of conversion

```
clr ADE               ; disable ADC interrupt
mov a,03H
mov SADC1,a           ; select fSYS/8 as A/D clock
set ADCEN
mov a,03h             ; setup ACERL to configure pin AN0
mov PBS0,a
mov a,20h
mov SADC0,a           ; enable and connect AN0 channel to A/D converter
:
Start_conversion:
clr START             ; high pulse on START bit to initiate conversion
set START             ; reset A/D
clr START             ; start A/D
clr ADF               ; clear ADC interrupt request flag
set ADE               ; enable ADC interrupt
set EMI               ; enable global interrupt
:
:
```

```
ADC_ISR:              ; ADC interrupt service routine
mov acc_stack,a       ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a    ; save STATUS to user defined memory
:
mov a, SADOL          ; read low byte conversion result value
mov adrl_buffer,a     ; save result to user defined register
mov a, SADOH          ; read high byte conversion result value
mov adrh_buffer,a     ; save result to user defined register
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a          ; restore STATUS from user defined memory
mov a,acc_stack       ; restore ACC from user defined memory
reti
```

Comparators

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



Comparator Operation

The devices contain two comparator functions which are used to compare two analog voltages and provide an output based on their input difference.

Any pull-high resistors connected to the shared comparator input pins will be automatically disconnected when the corresponding comparator functional pins are selected. As the comparator inputs approach their switching level, some spurious output signals may be generated on the comparator output due to the slow rising or falling nature of the input signals. This can be minimised by the hysteresis function. Ideally the comparator should switch at the point where the positive and negative inputs signals are at the same voltage level. However, unavoidable input offsets introduce some uncertainties here. The hysteresis function will also increases the switching offset value. The hysteresis window will be changed for different comparator response time selections. The comparator response time is shown in the comparator electrical characteristics.

Comparator Registers

Full control over each internal comparator is provided via two control registers, CMPnC and CMPnVOS. The comparator output is recorded via a bit in the respective control register, but can also be transferred out onto a shared I/O pin. Additional comparator functions include output polarity, response time and power down control.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CMPnC	—	CMPnEN	CnPOL	CMPnO	CNVTn1	CNVTn0	—	—
CMPnVOS	—	CnOFM	CnRSP	CnOF4	CnOF3	CnOF2	CnOF1	CnOF0

Comparator Registers List – n = 0 or 1

CMPnC Register

Bit	7	6	5	4	3	2	1	0
Name	—	CMPnEN	CnPOL	CMPnO	CNVTn1	CNVTn0	—	—
R/W	—	R/W	R/W	R	R/W	R/W	—	—
POR	—	0	0	0	0	0	—	—

Bit 7 Unimplemented, read as "0"

Bit 6 **CMPnEN**: Comparator n Enable Control
0: Disable
1: Enable
This is the Comparator n on/off control bit. If the bit is zero the comparator n will be switched off and no power consumed even if analog voltages are applied to its inputs. For power sensitive applications this bit should be cleared to zero if the comparator n is not used or before the devices enter the SLEEP or IDLE mode. Note that the comparator output will be set low when this bit is cleared to zero.

Bit 5 **CnPOL**: Comparator n output polarity Control
0: Output is not inverted
1: Output is inverted
This is the Comparator n polarity control bit. If the bit is zero then the comparator n output bit, CMPnO, will reflect the non-inverted output condition of the comparator n. If the bit is high the comparator n output bit will be inverted.

Bit 4 **CMPnO**: Comparator n output bit
If CnPOL=0,
0: Cn+ < Cn–
1: Cn+ > Cn–
If CnPOL=1,
0: Cn+ > Cn–
1: Cn+ < Cn–
This bit stores the Comparator n output bit. The polarity of the bit is determined by the voltages on the comparator n inputs and by the condition of the CnPOL bit.

Bit 3~2 **CNVTn1~CNVTn0**: Comparator n response time select
00: Response time = t_{RP0} (max.)
01: Response time = t_{RP1}
10: Response time = t_{RP2}
11: Response time = t_{RP3} (min.)
These bits are used to select the Comparator n response time. The response time is also determined by the overdriven voltage applied on the inputs except these bits. Refer to the comparator electrical characteristics for details.

Bit 1~0 Unimplemented, read as "0"

CMPnVOS Register

Bit	7	6	5	4	3	2	1	0
Name	—	CnOFM	CnRSP	CnOF4	CnOF3	CnOF2	CnOF1	CnOF0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	1	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **CnOFM**: Comparator n input offset calibration mode select
0: Normal operation mode
1: Offset calibration mode
This is the Comparator n offset calibration mode selection bit. If the bit is zero the comparator n will operate normally. The comparator n will enter the offset calibration mode if this bit is set to high.

Bit 5 **CnRSP**: Comparator n input offset calibration reference select
0: From Cn– input
1: From Cn+ input
This is the Comparator n offset calibration reference selection bit. When the comparator n is in the offset calibration mode, the calibration reference input can come from the comparator n inputs, Cn– or Cn+, determined by this bit. This bit is only available when the comparator n operates in offset calibration mode.

Bit 4~0 **CnOF4~CnOF0**: Comparator n input offset calibration control
0: From Cn– input
1: From Cn+ input
These bits are used to calibrate the input offset according to the selected reference input when the Comparator n is in the offset calibration mode. Refer to the corresponding "Offset Calibration Procedure" section for the detailed calibration procedure.

Offset Calibration Procedure

To operate in the input offset calibration mode for the Comparator n, the CnOFM bit should first be set to "1" followed by the reference input selection by configuring the CnRSP bit. For comparator n input offset calibration, the procedures are summarizedas as the following.

Step 1. Set CnOFM = 1 and configure the CnRSP bit, the comparator n will now operate in the comparator input offset calibration mode.

Step 2. Set CnOF [4:0] = 00000 and read the CMPnO bit.

Step 3. Increase the CnOF [4:0] value by 1 and then read the CMPnO bit.

If the CMPnO bit state has not changed, then repeat Step 3 until the CMPnO bit state has changed.

If the CMPnO bit state has changed, record the CnOF field value as CnVOS1 and then go to Step 4.

Step 4. Set CnOF [4:0] = 11111 and read the CMPnO bit.

Step 5. Decrease the CnOF [4:0] value by 1 and then read the CMPnO bit.

If the CMPnO bit state has not changed, then repeat Step 5 until the CMPnO bit state has changed.

If the CMPnO bit state has changed, record the CnOF field value as CnVOS2 and then go to Step 6.

Step 6. Restore the comparator input offset calibration value, CnVOS, into the CnOF [4:0] bit field. The offset calibration procedure is now finished and the input offset value is calculated by the following equation.

$$\text{CnVOS} = \frac{\text{CnVOS1} + \text{CnVOS2}}{2}$$

Serial Peripheral Interface – SPI

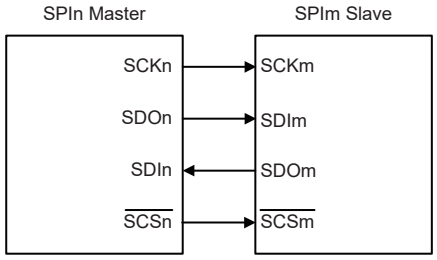
These devices contain two SPI functions, SPI0 and SPI1. The SPIn interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices, etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the programming requirements when communicating with external hardware devices.

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

SPI Interface Operation

The SPIn interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDIn, SDOn, SCKn and $\overline{\text{SCSn}}$. Pins SDIn and SDOn are the Serial Data Input and Serial Data Output lines. The SCKn pin is the Serial Clock line and $\overline{\text{SCSn}}$ is the Slave Select line. As the SPIn interface pins are pin-shared with other functions, the SPIn interface pins must first be selected by configuring the corresponding selection bits in the pin-shared function selection registers. The SPIn interface function is disabled or enabled using the SPInEN bit in the SPInC0 register. Communication between devices connected to the SPIn interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The master also controls the clock/signal. As the device only contains a single $\overline{\text{SCSn}}$ pin only one slave device can be utilised.

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

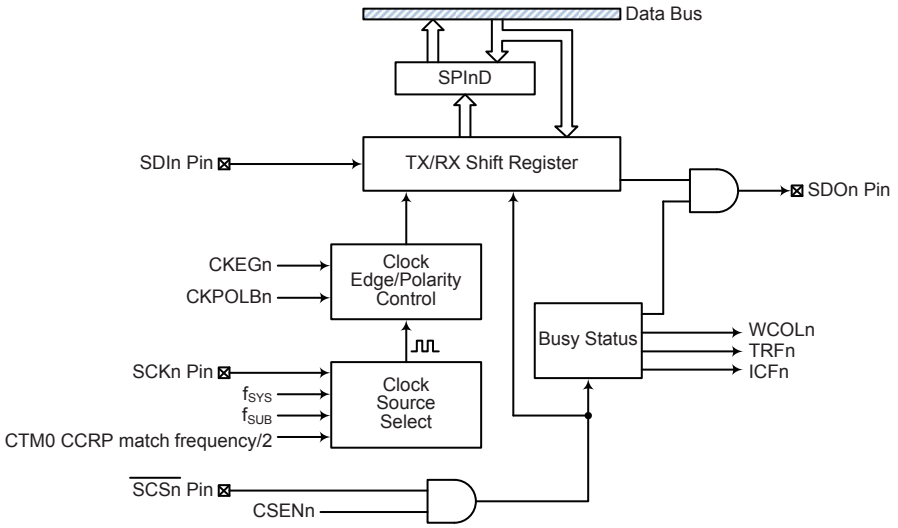


SPIn Master/Slave Connection

The SPIn Serial Interface function includes the following features:

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

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



SPIn Block Diagram

SPI Registers

There are three internal registers which control the overall operation of the SPIn interface. These are the SPInD data register and two control registers, SPInC0 and SPInC1.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SPInC0	SPInM2	SPInM1	SPInM0	—	—	—	SPInEN	ICFn
SPInC1	—	—	CKPOLBn	CKEGn	MLSn	CSENn	WCOLn	TRFn
SPInD	D7	D6	D5	D4	D3	D2	D1	D0

SPIn Registers List – n = 0 or 1

SPIn Data Register

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

- **SPInD 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

SPIn Control Register

There are also two control registers for the SPIn interface, SPInC0 and SPInC1. Register SPInC0 is used to control the enable/disable function and to set the data transmission clock frequency. Register SPInC1 is used for other control functions such as LSB/MSB selection, write collision flag, etc.

- **SPInC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SPInM2	SPInM1	SPInM0	—	—	—	SPInEN	ICFn
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	1	1	1	—	—	—	0	0

Bit 7~5 **SPInM2~SPInM0**: SPIn Master/Slave clock select
000: SPIn master mode with clock f_{SYS} /4
001: SPIn master mode with clock f_{SYS} /16
010: SPIn master mode with clock f_{SYS} /64
011: SPIn master mode with clock f_{SUB}
100: SPIn master mode with clock CTM CCRP match frequency/2
101: SPIn slave mode
11x: SPIn disabled

Bit 4~2 Unimplemented, read as "0"

Bit 1 **SPInEN**: SPIn Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SPIn interface. When the SPInEN bit is cleared to zero to disable the SPIn interface, the SDIn, SDOn, SCKn and $\overline{SCS}$n lines will lose the SPI function and the SPIn operating current will be reduced to a minimum value. When the bit is high the SPIn interface is enabled.

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

- **SPInC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CKPOLBn	CKEGn	MLSn	CSENn	WCOLn	TRFn
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 **CKPOLBn**: SPIn clock line base condition selection
0: The SCKn line will be high when the clock is inactive
1: The SCKn line will be low when the clock is inactive
The CKPOLBn bit determines the base condition of the clock line, if the bit is high, then the SCKn line will be low when the clock is inactive. When the CKPOLBn bit is low, then the SCKn line will be high when the clock is inactive.

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

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

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

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

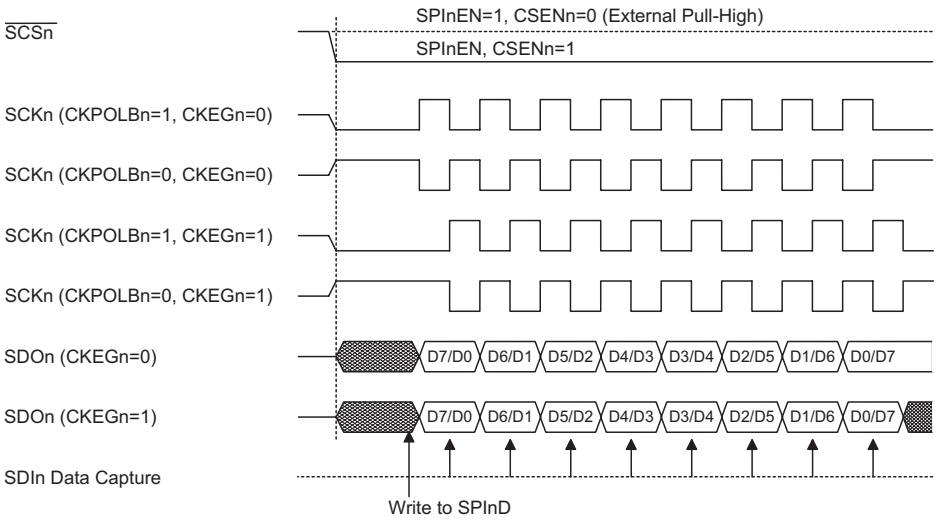
Bit 0 **TRFn**: SPIn Transmit/Receive complete flag
0: SPIn data is being transferred
1: SPIn data transfer is completed
The TRFn bit is the Transmit/Receive Complete flag and is set to 1 automatically when an SPIn data transfer is completed, but must cleared to 0 by the application program. It can be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

SPI Communication

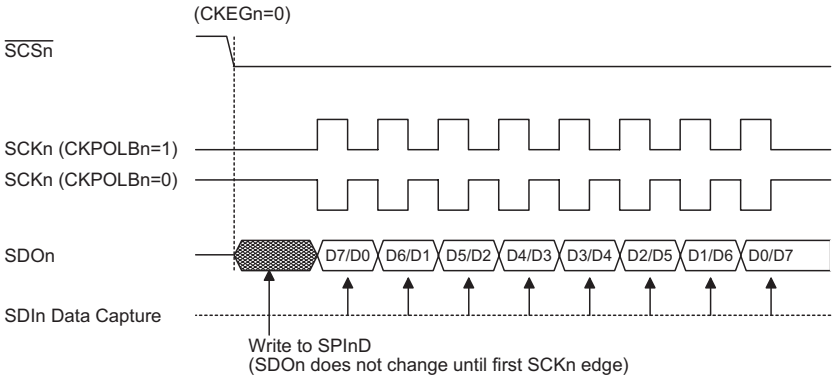
After the SPIn interface is enabled by setting the SPInEN bit high, then in the Master Mode, when data is written to the SPInD register, transmission/reception will begin simultaneously. When the data transfer is complete, the TRFn flag will be set automatically, but must be cleared using the application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

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

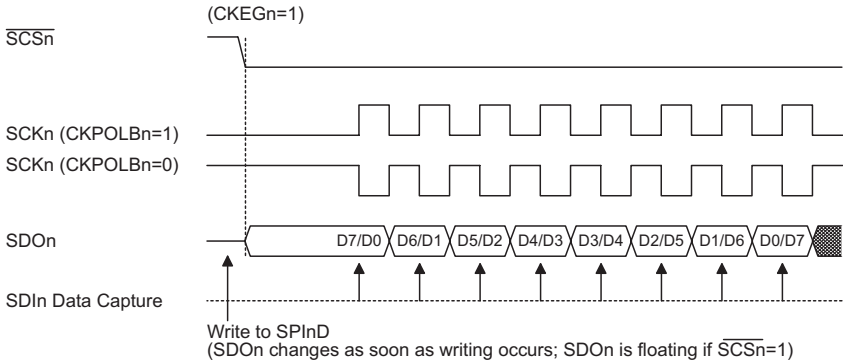
SIPn Master Mode



SPIn Slave Mode

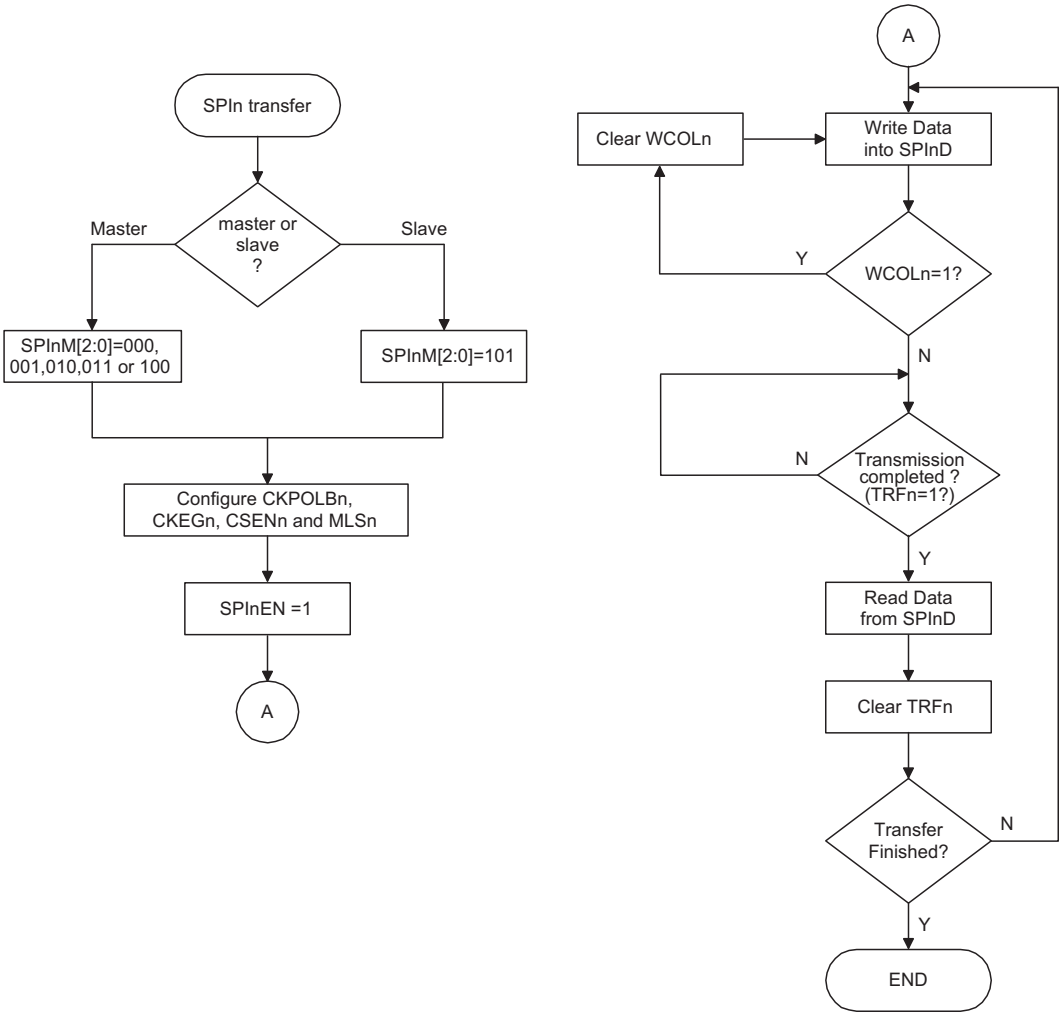


SPIn Slave Mode



Note: For SPIn Slave mode, if SPInEN=1 and CSENn=0, SPIn is always enabled and ignore the SCSn level.

SPIn Master/Slave Mode Timing Diagram



SPIn Transfer Control Flowchart

SPI Bus Enable/Disable

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

To Disable the SPIn bus SCKn, SDIn, SDOn, $\overline{SCSn}$ will become I/O pins or other pin-shared functions.

SPI Operation

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

The CSENn bit in the SPInC1 register controls the overall function of the SPIn interface. Setting this bit high will enable the SPIn interface by allowing the $\overline{SCSn}$ line to be active, which can then be used to control the SPIn interface. If the CSENn bit is low, the SPIn interface will be disabled and the $\overline{SCSn}$ line will be an I/O pin or other pin-shared functions and can therefore not be used for control of the SPIn interface. If the CSENn bit and the SPInEN bit in the SPInC0 register are set high, this will place the SDIn line in a floating condition and the SDOn line high. If in Master Mode the SCKn line will be either high or low depending upon the clock polarity selection bit CKPOLBn in the SPInC1 register. If in Slave Mode the SCKn line will be in a floating condition. If SPInEN is low then the bus will be disabled and $\overline{SCSn}$, SDIn, SDOn and SCKn pins will all become I/O pins or other pin-shared functions. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SPInD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode

- Step 1
  Select the clock source and Master mode using the SPInM2~SPInM0 bits in the SPInC0 control register
- Step 2
  Setup the CSENn bit and setup the MLSn bit to choose if the data is MSB or LSB shifted first, this must be same as the Slave device.
- Step 3
  Setup the SPInEN bit in the SPInC0 control register to enable the SPIn interface.
- Step 4
  For write operations: write the data to the SPInD register, which will actually place the data into the TXRX buffer. Then use the SCKn and $\overline{SCSn}$ lines to output the data. After this go to step 5.
  For read operations: the data transferred in on the SDIn line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPInD register.
- Step 5
  Check the WCOLn bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the TRFn bit or wait for a SPIn serial bus interrupt.
- Step 7
  Read data from the SPInD register.

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

Slave Mode

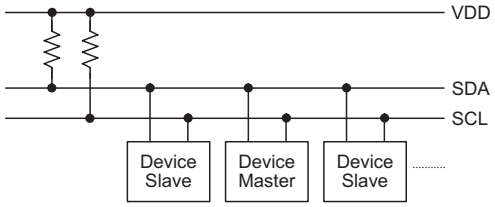
- Step 1
  Select the SPIn Slave mode using the SPInM2~SPInM0 bits in the SPInC0 control register
- Step 2
  Setup the CSENn bit and setup the MLSn bit to choose if the data is MSB or LSB shifted first, this setting must be the same with the Master device.
- Step 3
  Setup the SPInEN bit in the SPInC0 control register to enable the SPIn interface.
- Step 4
  For write operations: write the data to the SPInD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCKn and $\overline{\text{SCSn}}$ signal. After this, go to step 5.
  For read operations: the data transferred in on the SDIn line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPInD register.
- Step 5
  Check the WCOLn bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the TRFn bit or wait for a SPIn serial bus interrupt.
- Step 7
  Read data from the SPInD register.
- Step 8
  Clear TRFn.
- Step 9
  Go to step 4.

Error Detection

The WCOLn bit in the SPInC1 register is provided to indicate errors during data transfer. The bit is set by the SPIn serial Interface but must be cleared by the application program. This bit indicates a data collision has occurred which happens if a write to the SPInD register takes place during a data transfer operation and will prevent the write operation from continuing.

Inter-Integrated Circuit – I²C

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

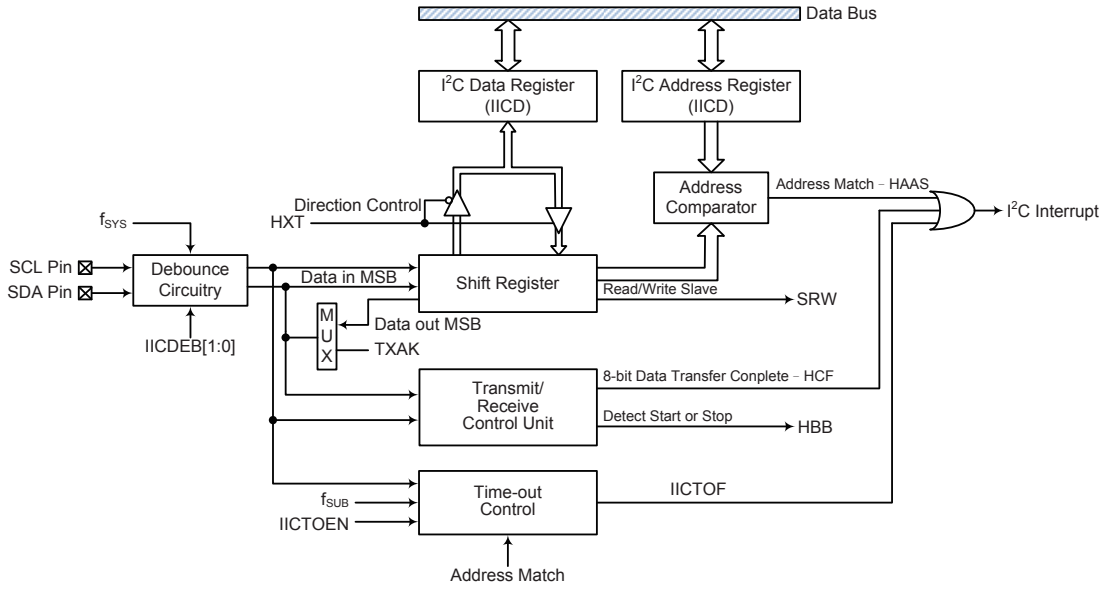


I²C Master Slave Bus Connection

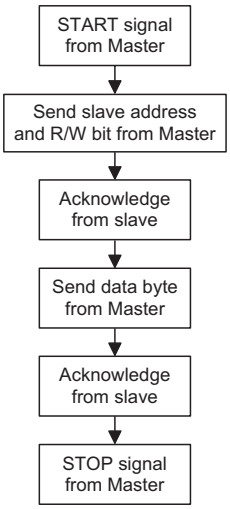
I²C interface Operation

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

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



I²C Block Diagram



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

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

I²C Minimum f_{SYS} Frequency

I²C Register

There are three control registers associated with the I²C bus, IICC0, IICC1, IICTOC, and one slave address register, IICA, together with one data register, IICD. The IICD register is used to store the data being transmitted and received on the I²C bus. Before the microcontroller writes data to the I²C bus, the actual data to be transmitted must be placed in the IICD register. After the data is received from the I²C bus, the microcontroller can read it from the IICD register. Any transmission or reception of data from the I²C bus must be made via the IICD register.

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

I²C Registers List

IICD 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.

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

IICA 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.

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	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 Undefined bit
The bit can be read or written by the application program.

I²C Control Registers

There are two control registers for the I²C interface, IICC0 and IICC1. The register IICC0 is used to control the enable/disable function and to set the data transmission clock frequency.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 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**: I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
1x: 4 system clock debounce
Note that the I²C 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 **IICEN**: I²C Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the I²C interface. When the IICEN bit is cleared to zero to disable the I²C interface, the SDA and SCL lines will lose their I²C function and the I²C operating current will be reduced to a minimum value. When the bit is high the I²C interface is enabled. If the IICEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 Unimplemented, read as "0"

- **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/W	R/W	R
POR	1	0	0	0	0	0	0	1

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

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

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

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

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

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

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

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

I²C Bus Communication

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the 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, 8-bit data transfer completion or 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 functional 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 IICE and corresponding Multi-Function interrupt enable bit of the interrupt control register to enable the IIC interrupt and Multi-function interrupt.

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, the completion of a data byte transfer or the I²C bus time-out occurrence. When a slave address is matched, the devices 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 Reae/Write Signal

The SRW bit in the IICC1 register defines whether the slave device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is “1” then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the SRW flag is “0” then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

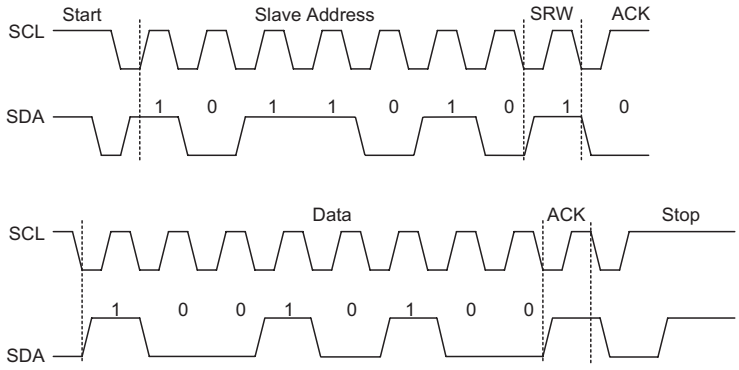
I²C Bus Slave Address Acknowledge Signal

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

I²C Bus Data and Acknowledge Signal

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

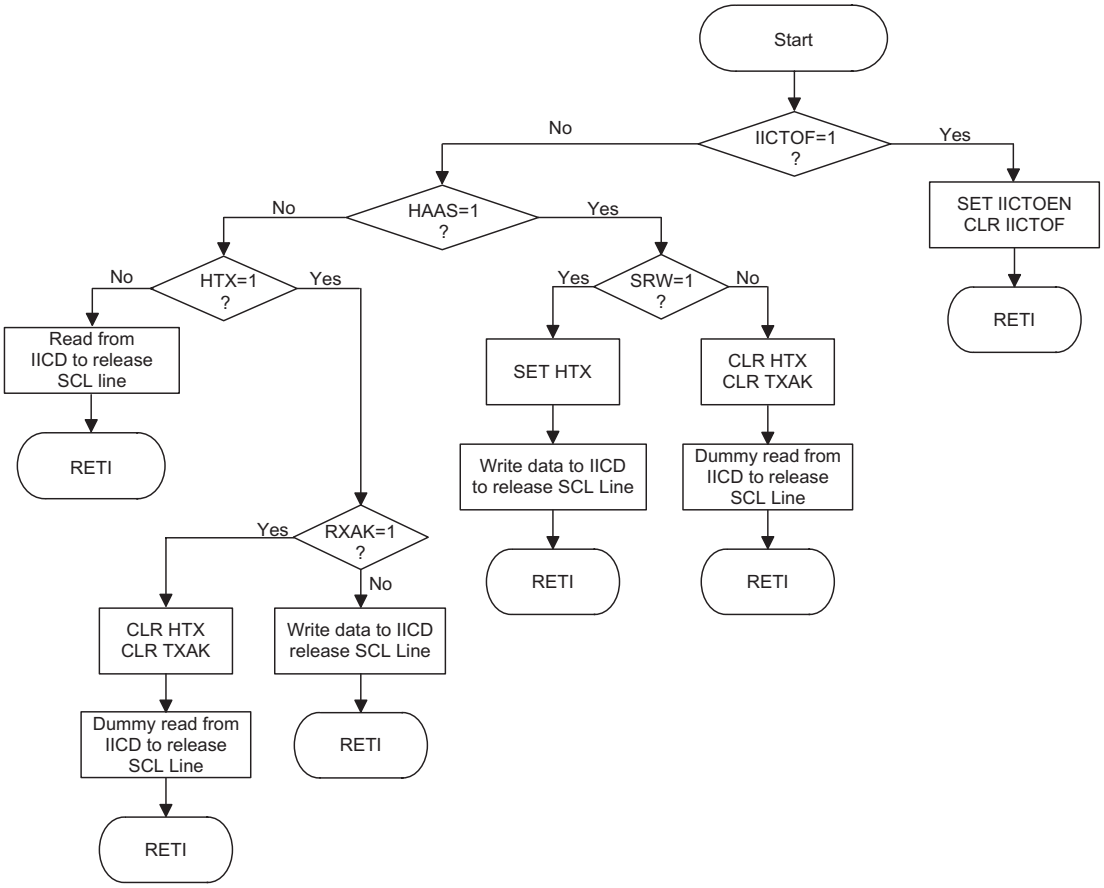


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

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

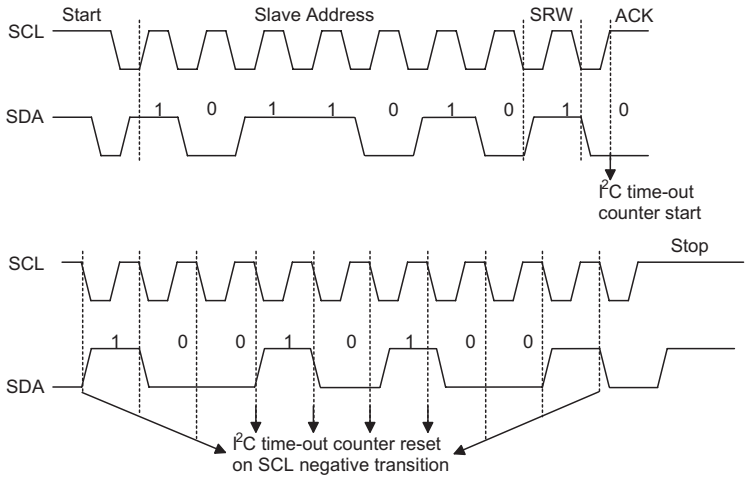
Note: *When a slave address is matched, the devices must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

I²C Communication Timing Diagram



I²C Time-out Control

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



I²C Time-out

When an I²C time-out counter overflow occurs, the counter will stop and the 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 I²C interrrupt vector. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

Register	After I²C Time-out
IICD, IICA, IICC0	No change
IICC1	Reset to POR condition

I²C Register after Time-out

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

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**: I²C Time-out control
0: Disable
1: Enable

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

Bit 5~0 **IICTOS5~IICTOS0**: I²C Time-out period selection
I²C Time-out clock source is $f_{SUB}/32$

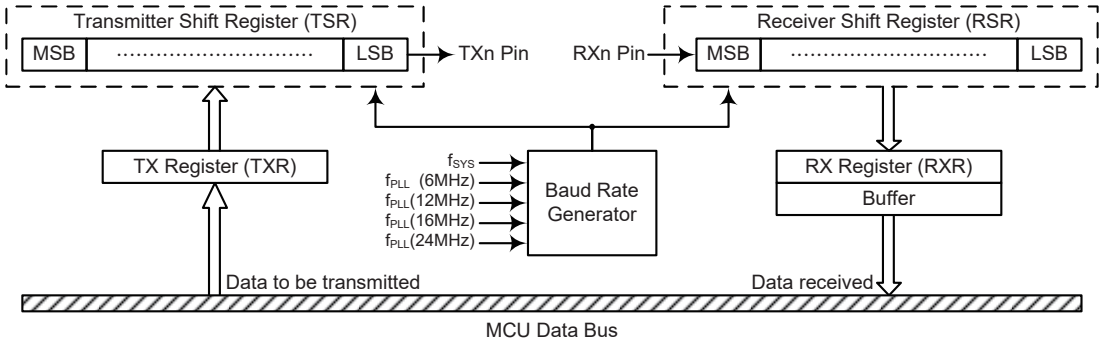
I²C Time-out period is equal to $(IICTOS[5:0] + 1) \times \frac{32}{f_{SUB}}$

UART Interface

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

The integrated UART function contains the following features:

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



UART Data Transfer Block Diagram – n = 0 or 1

UART External Pin

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

UART Data Transfer Scheme

The above diagram shows the overall data transfer structure arrangement for the UARTn interface. The actual data to be transmitted from the MCU is first transferred to the TXRn 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 TXn pin at a rate controlled by the Baud Rate Generator. Only the TXRn 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 UARTn is accepted on the external RXn 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 RXRn register, where it is buffered and can be manipulated by the application program. Only the TXRn register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.

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

UART Status and Control Registers

There are six control registers associated with the UARTn function. The UnSR, UnCR1 and UnCR2 registers control the overall function of the UARTn, while the BRGn register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXRn data registers. A clock selection register, URFC, is used to select the UARTn Baud Rate generator clock source.

Register Name	Bit							
	7	6	5	4	3	2	1	0
URFC	—	UR1FC2	UR1FC1	UR1FC0	—	UR0FC2	UR0FC1	UR0FC0
UnSR	PERRn	NFn	FERRn	OERRn	RIDLEn	RXIFn	TIDLEn	TXIFn
UnCR1	UARTENn	BNOn	PRENn	PRTn	STOPSn	TXBRKn	RX8n	TX8n
UnCR2	TXENn	RXENn	BRGHn	ADDENn	WAKEn	RIEn	TIIEn	TEIEn
BRGn	BRGn7	BRGn6	BRGn5	BRGn4	BRGn3	BRGn2	BRGn1	BRGn0
TXR_RXRn	TXRXn7	TXRXn6	TXRXn5	TXRXn4	TXRXn3	TXRXn2	TXRXn1	TXRXn0

UARTn Registers List – n = 0 or 1

TXR_RXRn Register

The TXR_RXRn register is the data register which is used to store the data to be transmitted on the TXn pin or being received from the RXn pin.

Bit	7	6	5	4	3	2	1	0
Name	TXRXn7	TXRXn6	TXRXn5	TXRXn4	TXRXn3	TXRXn2	TXRXn1	TXRXn0
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 **TXRXn7~TXRXn0**: UARTn Transmit/Receive Data bits

URFC Register

The URFC register is the clock selection register for the UARTn Baud Rate generator. The Baud Rate generator clock source can come from the system clock and various PLL output clocks.

Bit	7	6	5	4	3	2	1	0
Name	—	UR1FC2	UR1FC1	UR1FC0	—	UR0FC2	UR0FC1	UR0FC0
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~4 **UR1FC2~UR1FC0**: UART1 Baud Rate generator clock source select
000: f_{SYS}
001: f_{PLL} – 6MHz
010: f_{PLL} – 12MHz
011: f_{PLL} – 16MHz
100: f_{PLL} – 24MHz
101: Not used
110~111: f_{SYS}

Bit 3 Unimplemented, read as "0"

Bit 2~0 **UR0FC2~UR0FC0**: UART0 Baud Rate generator clock source select
000: f_{SYS}
001: f_{PLL} – 6MHz
010: f_{PLL} – 12MHz
011: f_{PLL} – 16MHz
100: f_{PLL} – 24MHz
101: Not used
110~111: f_{SYS}

UnSR Register

The UnSR register is the status register for the UARTn, which can be read by the program to determine the present status of the UARTn. All flags within the UnSR register are read only and further explanations are given below.

Bit	7	6	5	4	3	2	1	0
Name	PERRn	NFn	FERRn	OERRn	RIDLEn	RXIFn	TIDLEn	TXIFn
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

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

Bit 6 **NFn**: Noise flag
0: No noise is detected
1: Noise is detected
The NFn 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 UARTn has detected noise on the receiver input. The NFn flag is set during the same cycle as the RXIFn flag but will not be set in the case of as overrun. The NFn flag can be cleared by a software sequence which will involve a read to the status register UnSR followed by an access to the RXRn data register.

Bit 5 **FERRn**: Framing error flag
0: No framing error is detected
1: Framing error is detected
The FERRn 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 UnSR followed by an access to the RXRn data register.

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

Bit 3 **RIDLEn**: 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 RIDLEn bit is "1" indicating that the UARTn receiver is idle and the RXn pin stays in logic high condition.

Bit 2 **RXIFn**: Receive RXRn data register status
0: RXRn data register is empty
1: RXRn data register has available data
The RXIFn flag is the receive data register status flag. When this read only flag is "0", it indicates that the RXRn read data register is empty. When the flag is "1", it indicates that the RXRn read data register contains new data. When the contents of the shift register are transferred to the RXRn register, an interrupt is generated if RIEn=1 in the UnCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NFn, FERRn, and/or PERRn are set within the same clock cycle. The RXIFn flag is cleared when the UnSR register is read with RXIFn set, followed by a read from the RXRn register, and if the RXRn register has no data available.

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

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

UnCR1 Register

The UnCR1 register together with the UnCR2 register are the UARTn control registers that are used to set the various options for the UARTn function such as overall on/off control, parity control, data transfer bit length, etc. Further explanation on each of the bits is given below.

Bit	7	6	5	4	3	2	1	0
Name	UARTENn	BNOn	PRENn	PRTn	STOPSn	TXBRKn	RX8n	TX8n
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 **UARTENn**: UARTn function enable control
0: Disable UARTn; TXn and RXn pins are in a floating state.
1: Enable UARTn; TXn and RXn pins function as UARTn pins
The UARTENn bit is the UARTn enable bit. When this bit is equal to "0", the UARTn will be disabled and the RXn pin as well as the TXn pin will be set in a floating state. When the bit is equal to "1", the UARTn will be enabled and the TXn and RXn pins will function as defined by the TXENn and RXENn enable control bits. When the UARTn 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 UARTn is disabled, all error and status flags will be reset. Also the TXENn, RXENn, TXBRKn, RXIFn, OERRn, FERRn, PERRn and NFn bits will be cleared, while the

TIDLEn, TXIFn and RIDLEn bits will be set. Other control bits in UnCR1, UnCR2 and BRGn registers will remain unaffected. If the UARTn is active and the UARTENn 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 **BNOn**: 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 RX8n and TX8n will be used to store the 9th bit of the received and transmitted data respectively.

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

This bit is the parity function 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 **PRTn**: Parity type selection bit
0: Even parity for parity generator
1: Odd parity for parity generator

This bit is the parity type selection bit. When this bit is equal to 1, odd parity type will be selected. If the bit is equal to 0, then even parity type will be selected.

Bit 3 **STOPSn**: Number of stop bits selection
0: One stop bit format is used
1: Two stop bits format is used

This bit determines if one or two stop bits are to be used. When this bit is equal to "1", two stop bits format are used. If the bit is equal to "0", then only one stop bit format is used.

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

The TXBRKn bit is the Transmit Break Character bit. When this bit is equal to "0", there are no break characters and the TXn pin operats normally. When the bit is equal to "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 TXBRKn bit is reset.

Bit 1 **RX8n**: 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 RX8n. The BNOn bit is used to determine whether data transfes are in 8-bit or 9-bit format.

Bit 0 **TX8n**: 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 TX8n. The BNOn bit is used to determine whether data transfes are in 8-bit or 9-bit format.

UnCR2 Register

The UnCR2 register is the second of the UARTn control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation if the UARTn Transmitter and Receiver as well as enabling the various UARTn interrupt sources. The register also serves to control the baud rate speed, receiver wake-up function enable and the address detect function enable. Further explanation on each of the bits is given below.

Bit	7	6	5	4	3	2	1	0
Name	TXENn	RXENn	BRGHn	ADDENn	WAKEn	RIEn	TIIEn	TEIEn
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 **TXENn**: UARTn Transmitter enable control
0: UARTn Transmitter is disabled
1: UARTn Transmitter is enabled
The TXENn bit 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 TXn pin will be set in a floating state. If the TXENn bit is equal to "1" and the UARTENn bit is also equal to 1, the transmitter will be enabled and the TXn pin will be controlled by the UARTn. Clearing the TXENn bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TXn pin will be set in a floating state.

Bit 6 **RXENn**: UARTn Receiver enable control
0: UARTn Receiver is disabled
1: UARTn Receiver is enabled
The RXENn bit 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 receiver buffers will be reset. In this situation the RXn pin will be set in a floating state. If the RXENn bit is equal to "1" and the UARTENn bit is also equal to 1, the receiver will be enabled and the RXn pin will be controlled by the UARTn. Clearing the RXENn bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX pin will be set in a floating state.

Bit 5 **BRGHn**: Baud Rate speed selection
0: Low speed baud rate
1: High speed baud rate
The bit named BRGHn selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register, BRGn, controls the baud rate of the UARTn. If the bit is equal to 0, the low speed mode is selected.

Bit 4 **ADDENn**: Address detect function enable control
0: Address detection function is disabled
1: Address detection function is enabled
The bit named ADDENn is the address detection function enable control bit. When this bit is equal to 1, the address detection function is enabled. When it occurs, if the 8^{th} bit, which corresponds to RX7n if BNOn=0, or the 9^{th} bit, which corresponds to RX8n if BNOn=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 8^{th} or 9^{th} bit depending on the value of the BNOn bit. If the address bit known as the 8^{th} or 9^{th} bit of the received word is "0" with the address detection function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **WAKEn**: RXn pin falling edge wake-up function enable control
0: RXn pin wake-up function is disabled
1: RXn pin wake-up function is enabled
The bit enables or disables the receiver wake-up function. If this bit is equal to 1 and the device is in IDLE0 or SLEEP mode, a falling edge on the RXn pin will wake up the device. If this bit is equal to 0 and the device is in IDLE or SLEEP mode, any edge transitions on the RXn pin will not wake up the device.

Bit 2 **RIEn**: Receiver interrupt enable control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled
The bit enables or disables the receiver interrupt. If this bit is equal to 1 and when the receiver overrun flag OERRn or received data available flag RXIFn is set, the UARTn interrupt request flag will be set. If this bit is equal to 0, the UARTn interrupt request flag will not be influenced by the condition of the OERRn or RXIFn flags.

Bit 1 **TIIEn**: Transmitter Idle interrupt enable control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled
The bit enables or disables the transmitter idle interrupt. If this bit is equal to 1 and when the transmitter idle flag TIDLEn is set, due to a transmitter idle condition, the UARTn interrupt request flag will be set. If this bit is equal to 0, the UARTn interrupt request flag will not be influenced by the condition of the TIDLEn flag.

Bit 0 **TEIEn**: Transmitter Empty interrupt enable control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled
The bit enables or disables the transmitter empty interrupt. If this bit is equal to 1 and when the transmitter empty flag TXIFn is set, due to a transmitter empty condition, the UARTn interrupt request flag will be set. If this bit is equal to 0, the UARTn interrupt request flag will not be influenced by the condition of the TXIFn flag.

Baud Rate Generator

To setup the speed of the serial data communication, the UARTn function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRGn register and the second is the value of the BRGHn bit within the UnCR2 control register. The BRGHn bit decides, if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value in the BRGn register, N, which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the BRGn register and has a range of between 0 and 255.

UnCR2 BRGHn Bit	0	1
Baud Rate (BR)	$\frac{f_{SYS}}{[64(N+1)]}$	$\frac{f_{SYS}}{[16(N+1)]}$

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

BRGn Register

Bit	7	6	5	4	3	2	1	0
Name	BRGn7	BRGn6	BRGn5	BRGn4	BRGn3	BRGn2	BRGn1	BRGn0
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 **BRGn7~BRGn0**: Baud Rate values

By programming the BRGHn bit in the UnCR2 register which allows selection of the related formula described above and programming the required value in the BRGn register, the required baud rate can be setup.

Calculating the Baud Rate and Error Values

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

From the above table the desired baud rate $BR = \frac{f_{SYS}}{[64(N+1)]}$

Re-arranging this equation gives $N = \frac{f_{SYS}}{(BR \times 64)} - 1$

Giving a value for $N = \frac{4000000}{(4800 \times 64)} - 1 = 12.0208$

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

Therefore the error is equal to $\frac{4808-4800}{4800} = 0.16\%$

UART Setup and Control

For data transfer, the UARTn 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 UARTn hardware and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding BNOn, PRTn, PRENn and STOPSn bits in the UnCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the transmitter and receiver of the UARTn 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 UARTn function is controlled using the UARTENn bit in the UnCR1 register. If the UARTENn, TXENn and RXENn bits are set, then these two UARTn pins will act as normal TXn output pin and RXn input pin respectively. If no data is being transmitted on the TXn pin, then it will default to a logic high value.

Clearing the UARTENn bit will disable the TXn and RXn pins and these two pins will be used as an I/O or other pin-shared functional pin. When the UARTn function is disabled, the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UARTn will also reset the enable control, the error and status flags with bits TXENn, RXENn, TXBRKn, RXIFn, OERRn, FERRn, PERRn and NFn being cleared while bits TIDLEn, TXIFn and RIDLEn will be set. The remaining control bits in the UnCR1, UnCR2 and BRGn registers will remain unaffected. If the UARTENn bit in the UnCR1 register is cleared while the UARTn is active, then all pending transmissions and receptions will be immediately suspended and the UARTn will be reset to a condition as defined above. If the UARTn is then subsequently re-enabled, it will restart again in the same configuration.

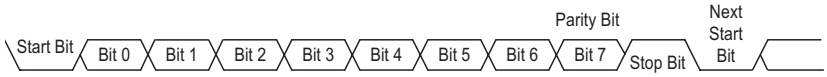
Data, Parity and Stop Bit Selection

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 UnCR1 register. The BNOn bit controls the number of data bits which can be set to either 8 or 9. The PRTn bit controls the choice if odd or even parity. The PRENn bit controls the parity on/off function. The STOPSn bit decides whether one or two stop bits are to be used. The following table shows various formats for data transmission. The address detect mode control bit identifies the frame as an address character. The number of stop bits, which can be either one or two, is independent of the data length.

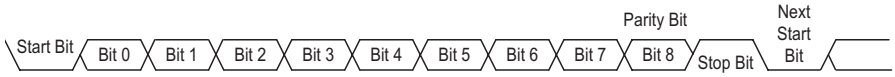
Start Bit	Data Bits	Address Bits	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
Example of 9-bit Data Formats				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

Transmitter Receiver Data Format

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



8-Bit Data Format



9-Bit Data Format

UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNOn bit in the UnCR1 register. When BNOn 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 TX8n bit in the UnCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSRn, whose data is obtained from the transmit data register, which is known as the TXRn register. The data to be transmitted is loaded into this TXRn register by the application program. The TSRn 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 TSRn can then be loaded with new data from the TXRn register, if it is available. It should be noted that the TSRn 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 TXENn bit is set, but the data will not be transmitted until the TXRn 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 TXRn register, after which the TXENn bit can be set. When a transmission of data begins, the TSRn is normally empty, in which case a transfer to the TXRn register will result in an immediate transfer to the TSRn. If during a transmission the TXENn bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TXn output pin can then be configured as the I/O or other pin-shared function.

Transmitting Data

When the UARTn is transmitting data, the data is shifted on the TXn pin from the shift register, with the least significant bit LSB first. In the transmit mode, the TXRn 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 TX8n bit in the UnCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNOn, PRTn, PRENn and STOPSn bits to define the required word length, parity type and number of stop bits.
- Setup the BRGn register to select the desired baud rate.
- Set the TXENn bit to ensure that the UARTn transmitter is enabled and the TXn pin is used as a UARTn transmitter pin.
- Access the UnSR register and write the data that is to be transmitted into the TXRn register. Note that this step will clear the TXIFn bit.

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

1. A UnSR register access
2. A TXRn register write execution

The read-only TXIFn flag is set by the UARTn hardware and if set indicates that the TXRn register is empty and that other data can now be written into the TXRn register without overwriting the previous data. If the TEIEn bit is set, then the TXIFn flag will generate an interrupt. During a data transmission, a write instruction to the TXRn register will place the data into the TXRn 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 TXRn register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIFn bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLEn bit will be set. To clear the TIDLEn bit the following software sequence is used:

1. A UnSR register access
2. A TXRn register write execution

Note that both the TXIFn and TIDLEn bits are cleared by the same software sequence.

Transmitting Break

If the TXBRKn bit is set, then the break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13xN “0” bits, where N=1, 2, etc. If a break character is to be transmitted, then the TXBRKn 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 TXBRKn 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 TXBRKn bit, the transmitter will finish transmitting the last break character and subsequently send out one or two stop bits. The automatic logic high at the end of the last break character will ensure that the start bit of the next frame is recognized.

UART Receiver

The UARTn is capable of receiving word lengths of either 8 or 9 bits can be selected by programming the BNOn bit in the UnCR1 register. When BNOn bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, will be stored in the RX8n bit in the UnCR1 register. At the receiver core lies the Receiver Shift Register more commonly known as the RSRn. The data which is received on the RXn 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 RXn pin is sampled for the stop bit, the received data in RSRn is transferred to the receive data register, if the register is empty. The data which is received on the external RXn input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RXn pin. It should be noted that the RSRn 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 UARTn receiver is receiving data, the data is serially shifted in on the external RXn input pin to the shift register, with the least significant bit LSB first. The RXRn register is a two byte deep FIFO data buffer, where two bytes can be held in the FIFO while the 3rd byte can continue to be received. Note that the application program must ensure that the data is read from RXRn before the 3rd byte has been completely shifted in, otherwise the 3rd byte will be discarded and an overrun error OERRn will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNOn, PRTn, PRENn and STOPSn bits to define the required word length, parity type and number of stop bits.
- Setup the BRGn register to select the desired baud rate.
- Set the RXENn bit to ensure that the UARTn receiver is enabled and the RXn pin is used as a UARTn 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 RXIFn bit in the UnSR register will be set then RXRn register has data available, at least one more character can be read.
- When the contents of the shift register have been transferred to the RXRn register and if the RIEn bit is set, then an interrupt will be generated.
- If during reception, a frame error, noise error, parity error or an overrun error has been detected, then the error flags can be set.

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

1. A UnSR register access

2. A RXRn register read execution

Receiving Break

Any break character received by the UARTn 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 BNOn and STOPSn 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 BNOn and STOPSn. The RXIFn bit is set, FERRn is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLEn bit is set. If a long break signal has been detected and the receiver has received a start bit, the data bits and the invalid stop bit, which sets the FERRn flag, 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. A break is regarded as a character that contains only zeros with the FERRn flag set. The break character will be loaded into the buffer and no further data will be received until stop bits are received. It should be noted that the RIDLEn read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UARTn registers will result in the following:

- The framing error flag, FERRn, will be set.
- The receive data register, RXRn, will be cleared.
- The OERRn, NFn, PERRn, RIDLEn or RXIFn 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 UnSR register, otherwise known as the RIDLEn flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the RIDLEn flag will have a high value, which indicates the receiver is in an idle condition.

Receiver Interrupt

The read only receive interrupt flag, RXIFn, in the UnSR register is set by an edge generated by the receiver. An interrupt is generated if RIEn=1, when a word is transferred from the Receive Shift Register, RSRn, to the Receive Data Register, RXRn. An overrun error can also generate an interrupt if RIEn=1.

Managing Receiver Errors

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

Overrun Error – OERR

The RXRn register is composed of a two byte deep FIFO data buffer, where two bytes can be held in the FIFO register, while a 3th byte can continue to be received. Before the 3th byte has been entirely shifted in, the data should be read from the RXRn register. If this is not done, the overrun error flag OERRn will be consequently indicated.

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

- The OERRn flag in the UnSR register will be set.
- The RXRn contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIEn bit is set.

The OERRn flag can be cleared by an access to the UnSR register followed by a read to the RXRn 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, NFn, in the UnSR register will be set on the rising edge of the RXIFn bit.
- Data will be transferred from the shift register to the RXRn register.
- No interrupt will be generated. However this bit rises at the same time as the RXIFn bit which itself generates an interrupt.

Note that the NFn flag is reset by a UnSR register read operation followed by an RXRn register read operation.

Framing Error – FERR

The read only framing error flag, FERRn, in the UnSR 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 FERRn flag will be set. The FERRn flag is buffered along with the received data and is cleared in any reset.

Parity Error – PERR

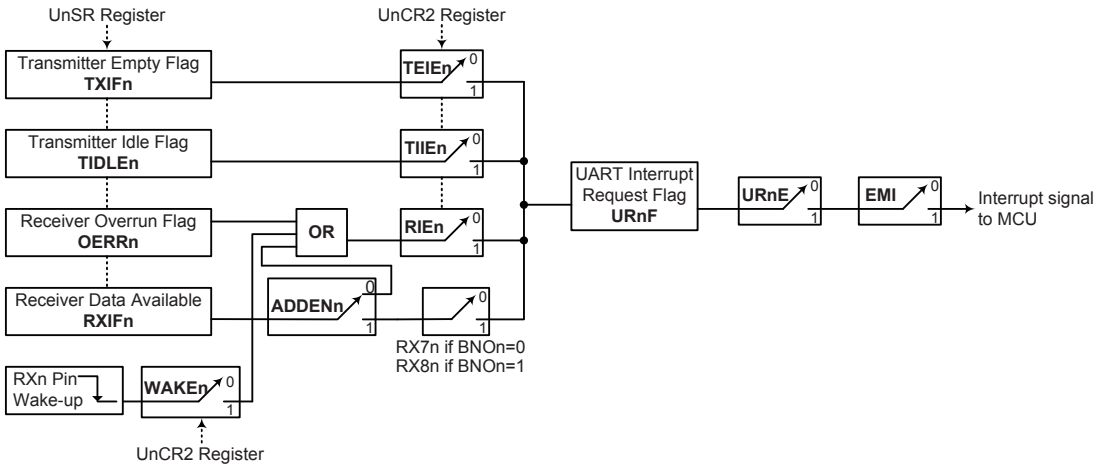
The read only parity error flag, PERRn, in the UnSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity function is enabled, PRENn=1, and if the parity type, odd or even, is selected. The read only PERRn flag is buffered along with the received data bytes. It is cleared on any reset, it should be noted that the FERRn and PERRn flags are buffered along with the corresponding word and should be read before reading the data word.

UART Interrupt Structure

Several individual UARTn conditions can generate a UARTn interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RXn pin wake-up. When any of these conditions are created, if its corresponding interrupt control is enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding UnSR register flags which will generate a UARTn interrupt if its associated interrupt enable control bit in the UnCR2 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 UARTn interrupt sources.

The address detect condition, which is also a UARTn interrupt source, does not have an associated flag, but will generate a UARTn interrupt when an address detect condition occurs if its function is enabled by setting the ADDENn bit in the UnCR2 register. An RXn pin wake-up, which is also a UARTn interrupt source, does not have an associated flag, but will generate a UARTn interrupt if the microcontroller is woken up from IDLE0 or SLEEP mode by a falling edge on the RXn pin, if the WAKEn and RIEn bits in the UnCR2 register are set. Note that in the event of an RXn wake-up interrupt occurring, there will be a certain period of delay, commonly known as the System Start-up Time, for the oscillator to restart and stabilize before the system resumes normal operation.

Note that the UnSR 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 UARTn, the details of which are given in the UARTn register section. The overall UARTn 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 UARTn module is masked out or allowed.



UARTn Interrupt Structure

Address Detect Mode

Setting the Address Detect function enable control bit, ADDENn, in the UnCR2 register, enables this special function. If this bit is set to 1, then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIFn flag. If the ADDENn bit is equal to 1, then when the data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the related interrupt enable control bit and the EMI bit of the microcontroller must also be enabled for correct interrupt generation. The highest address bit is the 9th bit if the bit BNOn=1 or the 8th bit if the bit BNOn=0. If the highest 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 ADDENn bit is equal to 0, then a Receive Data Available interrupt will be generated each time the RXIFn flag is set, irrespective of the data last but status. The address detection and parity functions are mutually exclusive functions. Therefore, if the address detect function is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity function enable bit PRENn to zero.

ADDENn	Bit 9 if BNOn=1 Bit 8 if BNOn=0	UARTn Interrupt Generated
0	0	√
	1	√
1	0	X
	1	√

ADDENn Bit Function

UART Power Down and Wake-up

When the MCU system clock is switched off, the UARTn will cease to function. If the MCU executes the "HALT" instruction and switches off the system clock while a transmission is still in progress, then the transmission will be paused until the UARTn clock source derived from the microcontroller is activated. In a similar way, if the MCU executes the "HALT" instruction and switches off the system clock while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP Mode, note that the UnSR, UnCR1, UnCR2, transmit and receive registers, as well as the BRGn register will not be affected. It is recommended to make sure first that the UARTn data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UARTn function contains a receiver RXn pin wake-up function, which is enabled or disabled by the WAKEn bit in the UnCR2 register. If this bit, along with the UARTn enable bit, UARTENn, the receiver enable bit, RXENn and the receiver interrupt bit, RIEn, are all set before the MCU enters the IDLE0 or SLEEP Mode, then a falling edge on the RXn pin will wake up the MCU from the IDLE0 or SLEEP Mode. 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 RXn pin will be ignored.

For a UARTn wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UARTn interrupt enable bit, URnE, must be set. If the EMI and URnE 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 UARTn interrupt will not be generated until after this time has elapsed.

USB Interface

The USB interface is a 4-wire serial bus that allows communication between a host device and up to 127 max peripheral devices on the same bus. A token based protocol method is used by the host device for communication control. Other advantages of the USB bus include live plugging and unplugging and dynamic device configuration. As the complexity of USB data protocol does not permit comprehensive USB operation information to be provided in this datasheet, the reader should therefore consult other external information for a detailed USB understanding.

The devices include a USB interface function allowing for the convenient design of USB peripheral products.

Power Plane

There are three power planes for these devices and they are USB SIE VDD, VDDIO and the MCU VDD.

For the USB SIE VDD will supply all circuits related to USB SIE and be sourced from pin "UBUS". Once the USB is removed from the USB and there is no power in the USB BUS, the USB SIE circuit is no longer operational.

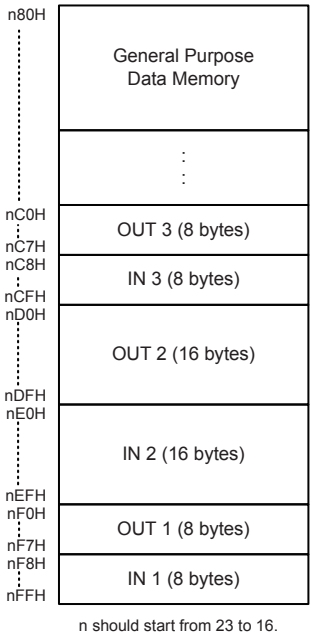
For the Port A and Port B, the power can be supplied by the VDD, V33O or VDDIO pin selected using the IOVC1 and IOVC0 bits in the SYSC register.

The PE0 pin is pin-shared with VDDIO pin. The VDDIO function can be selected by the corresponding pin-shared function selection bits. The PE1 pin is pin-shared with UBUS pin and it's "input" only.

USB Interface Operation

To communicate with an external USB host, the internal USB module has the external pins known as UDP and UDN along with the 3.3V regulator output V33O. A Serial Interface Engine(SIE) decodes the incoming USB data stream and transfers it to the correct endpoint buffer memoryknown as the FIFO. The USB module has 8 endpoints, EP0 ~ EP7, and the FIFO size for each endpoint except endpoint 0 can respectively be configured using the UFC0 and UFC1 registers by application programs. All endpoints except endpoint 0 can be configured to have 8, 16, 32 or 64 bytes together with the FIFOn register as the FIFO size. The endpoint 0 has 8-byte FIFO size. The endpoint 0 supports the Control transfer while the endpoint 1 ~ endpoint 7 support the Interrupt or Bulk transfer.

As the USB FIFO is assigned from the last bank of the General Purpose Data Memory and has a start address to the upper address, dependent on the FIFO size, if the corresponding data RAM bank is used for both general purpose RAM and the USB FIFO, special care should be taken that the RAM EQU definition should not overlap with the USB FIFO RAM address. The USB FIFO size and definition for IN/OUT control depends upon the UFC0, UFC1, UFIEN and UFOEN registers.



USB FIFO Size Configuration Example

USB Interface Registers

The USB function control is implemented using a series of registers. A series of status registers provide the user with the USB data transfer situation as well as any error conditions. The USB contains its own independent interrupt which can be used to indicate when the USB FIFOs are accessed by the host device or a change of the USB operating conditions including the USB suspend mode, resume event or USB reset occurs.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SYSC	CLK_ADJ	USBDIS	RUBUS	D4	D3	D2	IOVC1	IOVC0
USB_STAT	PS2_CKO	PS2_DAO	PS2_CKI	PS2_DAI	SE1	SE0	PU	ESD
UINT	EP7EN	EP6EN	EP5EN	EP4EN	EP3EN	EP2EN	EP1EN	EP0EN
USC	URD	SELPS2	PLL	SELUSB	RESUME	URST	RMWK	SUSP
UESR	EP7F	EP6F	EP5F	EP4F	EP3F	EP2F	EP1F	EP0F
UCC	RCTRL	SYSCLK	FSYS16MHZ	SUSP2	USBCKEN	EPS2	EPS1	EPS0
AWR	AD6	AD5	AD4	AD3	AD2	AD1	AD0	WKEN
STLI	STLI7	STLI6	STLI5	STLI4	STLI3	STLI2	STLI1	STLI0
STLO	STLO7	STLO6	STLO5	STLO4	STLO3	STLO2	STLO1	—
SIES	NMI	CRCF	—	NAK	IN	OUT	ERR	ASET
MISC	LEN0	READY	SETCMD	D4	E3IDF	CLEAR	TX	REQUEST
PLLC	—	—	—	—	—	—	PLLF	—
UFIEN	SETI7	SETI6	SETI5	SETI4	SETI3	SETI2	SETI1	FIFO_DEF
UFOEN	SETO7	SETO6	SETO5	SETO4	SETO3	SETO2	SETO1	DATATG
UFC0	E3FS1	E3FS0	E2FS1	E2FS0	E1FS1	E1FS0	—	—
UFC1	E7FS1	E7FS0	E6FS1	E6FS0	E5FS1	E5FS0	E4FS1	E4FS0
FIFO0	D7	D6	D5	D4	D3	D2	D1	D0
FIFO1	D7	D6	D5	D4	D3	D2	D1	D0
FIFO2	D7	D6	D5	D4	D3	D2	D1	D0
FIFO3	D7	D6	D5	D4	D3	D2	D1	D0
FIFO4	D7	D6	D5	D4	D3	D2	D1	D0
FIFO5	D7	D6	D5	D4	D3	D2	D1	D0
FIFO6	D7	D6	D5	D4	D3	D2	D1	D0
FIFO7	D7	D6	D5	D4	D3	D2	D1	D0

USB Interface Registers List

SYSC Register

Bit	7	6	5	4	3	2	1	0
Name	CLK_ADJ	USBDIS	RUBUS	D4	D3	D2	IOVC1	IOVC0
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 **CLK_ADJ**: PLL clock automatic adjustment control
Discribed elsewhere.

Bit 6 **USBDIS**: USB SIE function control
0: SIE enabled
1: SIE disabled
This bit is used to control the USB SIE function. When this bit is set to 1, the USB SIE function will be disabled.

Bit 5 **RUBUS**: UBUS pin pull low function control
0: Enable
1: Disable

Bit 4~2 **D4~D2**: Reserved data bits, should be kept low and cannot be modified.

Bit 1~0 **IOVC1~IOVC0**: I/O port A and B power select
00: From VDD
01: From V33O (3.3V)
10: From VDDIO
11: From VDD
These bits are used to select the PA and PB pins supply power. The SELUSB bit in the USC register should first be set to 1 before the V33O power can be selected as the PA and PB pins supply power. If the PE0 pin is configured as the VDDIO pin function, then the VDDIO input voltage can be used as the PA and PB pins supply power. No matter which one here is selected as the PA and PB pins supply power, the selected power voltage should be equal to or less than the VDD voltage.

USB_STAT Register

Bit	7	6	5	4	3	2	1	0
Name	PS2_CKO	PS2_DAO	PS2_CKI	PS2_DAI	SE1	SE0	PU	ESD
R/W	W	W	R	R	R/W	R/W	R/W	R/W
POR	1	1	0	0	0	0	0	0

Bit 7 **PS2_CKO**: Output for driving the UDP/GPIO1 pin when operating in the 3D PS2 mode.

Bit 6 **PS2_DAO**: Output for driving the UDN/GPIO0 pin when operating in the 3D PS2 mode.

Bit 5 **PS2_CKI**: UDP/GPIO1 Input.

Bit 4 **PS2_DAI**: UDN/GPIO0 Input.

Bit 3 **SE1**: USB bus SE1 noise indication
This bit is used to indicate that the SIE has detected a SE1 noise on the USB bus. This bit is set by SIE and cleared by software.

Bit 2 **SE0**: USB bus SE0 noise indication
This bit is used to indicate that the SIE has detected a SE1 noise on the USB bus. This bit is set by SIE and cleared by software.

Bit 1 **PU**: UDP/UDN pins pull-high function control
0: Disable – no internal pull-high resistor
1: Enable – internal 600kΩ pull-high resistor on UDP/UDN pins

Bit 0 **ESD**: ESD issue adjustment control
0: No operation
1: ESD adjustment
This bit will be set to 1 when there is an ESD issue. It is set by SIE and cleared by software.

UINT Register

Bit	7	6	5	4	3	2	1	0
Name	EP7EN	EP6EN	EP5EN	EP4EN	EP3EN	EP2EN	EP1EN	EP0EN
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 **EP7EN**: Endpoint 7 interrupt enable control
0: Disable
1: Enable

Bit 6 **EP6EN**: Endpoint 6 interrupt enable control
0: Disable
1: Enable

Bit 5 **EP5EN**: Endpoint 5 interrupt enable control
0: Disable
1: Enable

Bit 4 **EP4EN**: Endpoint 4 interrupt enable control
0: Disable
1: Enable

Bit 3 **EP3EN**: Endpoint 3 interrupt enable control
0: Disable
1: Enable

Bit 2 **EP2EN**: Endpoint 2 interrupt enable control
0: Disable
1: Enable

Bit 1 **EP1EN**: Endpoint 1 interrupt enable control
0: Disable
1: Enable

Bit 0 **EP0EN**: Endpoint 0 interrupt enable control
0: Disable
1: Enable

USC Register

Bit	7	6	5	4	3	2	1	0
Name	URD	SELPS2	PLL	SELUSB	RESUME	URST	RMWK	SUSP
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R
POR	1	0	0	0	0	0	0	0

Bit 7 **URD**: USB reset signal reset function control
0: USB reset signal can not reset MCU
1: USB reset signal will reset MCU

Bit 6 **SELPS2**: PS2 mode select
0: PS2 mode is not selected
1: PS2 mode is selected
This selection bit is available when the SELUSB selection bit is equal to 0. When the SELUSB selection bit is set low and the SELPS2 bit is set high, the PS2 function is selected and the pin-shared pins, UDP/GPIO1 and UDN/GPIO0, will become the GPIO1 and GPIO0 general purpose I/O functions which can be used to be the DATA and CLK pins for the PS2. However, when the SELPS2 bit is set low, the PS2 function will be disabled and the GPIO1 and GPIO0 output function will also be disabled.

Bit 5 **PLL**: PLL circuit enable control
Discribed elsewhere.

Bit 4 **SELUSB**: USB analog function and V33O enable control
0: USB analog function and the V33O is disabled
1: USB analog function and the V33O is enabled
When the SELUSB bit is set high, the USB analog function and V33O functions will both be enabled and the pin-shared pins, UDP/GPIO1 and UDN/GPIO0, will become the UDP and UDN USB functional pins. When this bit is set low, the USB analog function will be disabled, the V33O output will be floating and the UDP/GPIO1 and UDN/GPIO0 pins will become GPIO1 and GPIO0 functional pins.

Bit 3 **RESUME**: USB resume indication
0: Resume signal is not asserted or USB device has left the suspend mode
1: Resume signal is asserted and USB device is going to leave the suspend mode
When the resume event occurs, this bit will be set high by SIE and then an interrupt will also be generated to wake up the MCU. In order to detect the suspend state, the MCU should set the USBCKEN bit to 1 and clear the SUSP2 bit to 0. When the USB device leaves the suspend mode, the SUSP bit will be cleared to 0 and then the RESUME bit will also be cleared to 0. The resume signal which causes the MCU to wake up should be noted and taken into consideration when the MCU is detecting the suspend mode.

Bit 2 **URST**: USB reset indication
0: No USB reset event occurs
1: USB reset event occurs
This bit is set and cleared by the SIE. When the URST bit is set high, it indicates that a USB reset event has occurred and a USB interrupt will be generated.

Bit 1 **RMWK**: USB remote wake-up command
0: No USB remote wake-up command initiated
1: Initiate USB remote wake-up command
The RMEK bit is set to 1 by the MCU to force the USB host leaving the suspend mode. Setting the RMWK bit to 1 will initiate a remote wake-up command. The RMWK bit should be kept high for at least 2μs to make sure that the remote wake-up command is accepted by the SIE.

Bit 0 **SUSP**: USB suspend indication
0: USB leaves the suspend mode
1: USB enters the suspend mode
This bit is read only and set to 1 by the SIE to indicate that the USB has entered the suspend mode. The corresponding interrupt will also be generated when the SUSP bit changes from low to high.

SELUSB	SELPS2	USB and PS2 mode description
0	0	1. No mode supported 2. V33O pin not output and it will floating 3. UDN/GPIO0 and UDP/GPIO1 pins cann't output
0	1	1. PS2 mode 2. V33 O pin output VDD 3. UDN/GPIO0 and UDP/GPIO1 pins will become the GPIO0 and GPIO1 pins, which can output by firmware
1	x	1. USB mode 2. V33O output 3.3V 3. UDN/GPIO0 and UDP/GPIO1 pins will become the UDN and UDP pins

UESR Register

The UESR register is the USB endpoint interrupt status register and is used to indicate which endpoint is accessed and to select the USB bus. The endpoint request flags, EPxF, are used to indicate which endpoints are accessed. If an endpoint is accessed, the related endpoint request flag will be set to "1" and the USB interrupt will occur if the USB interrupt is enabled and the stack is not full. When the active endpoint request flag is serviced, the endpoint request flag has to be cleared to "0" by software.

Bit	7	6	5	4	3	2	1	0
Name	EP7F	EP6F	EP5F	EP4F	EP3F	EP2F	EP1F	EP0F
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 **EP7F~EP0F**: Endpoint 7~Endpoint 0 access interrupt request flag – EPnF
0: Endpoint n is not accessed
1: Endpoint n has been accessed

UCC Register

Bit	7	6	5	4	3	2	1	0
Name	RCTRL	SYSCLK	FSYS16MHZ	SUSP2	USBCKEN	EPS2	EPS1	EPS0
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 **RCTRL**: 7.5kΩ resistor between UDP and UBUS connection control
0: Disable – no 7.5kΩ resistor is connected between UDP and UBUS lines
1: Enable – 7.5kΩ resistor is connected between UDP and UBUS lines

Bit 6 **SYSCLK**: System clock frequency indicator
0: 12MHz clock is used
1: 6MHz clock is used
This bit is used to specify the system clock oscillator used by the MCU. If a 6MHz crystal or resonator is used for the device, this bit should be set to 1. If a 12MHz crystal or resonator is used, then this bit should be set to 0. If the 12MHz HIRC is used, then this bit must be set to 0.

Bit 5 **FSYS16MHZ**: MCU System clock select
0: System clock comes from HXT or HIRC
1: System clock comes from PLL output – 16MHz

Bit 4 **SUSP2**: USB suspend mode current reduction control
0: Current reduction is disabled in suspend mode
1: Current reduction is enabled in suspend mode
The current can be reduced to meet the USB standard specification if this bit is set to 1 when entering the suspend mode.

Bit 3 **USBCKEN**: USB clock enable control
0: Disable
1: Enable

Bit 2~0 **UPS2~UPS0**: Endpoint FIFO access selection
000: Endpoint 0 FIFO is selected
001: Endpoint 1 FIFO is selected
010: Endpoint 2 FIFO is selected
011: Endpoint 3 FIFO is selected
100: Endpoint 4 FIFO is selected
101: Endpoint 5 FIFO is selected
110: Endpoint 6 FIFO is selected
111: Endpoint 7 FIFO is selected

AWR Register

The AWR register contains the USB device address and the remote wake up function control bit. The initial value of the USB device address is "00H". The address value extracted from the USB host command is immediately loaded into this register or not is determined by the ASET bit in the SIES register.

Bit	7	6	5	4	3	2	1	0
Name	AD6	AD5	AD4	AD3	AD2	AD1	AD0	WKEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~1 **AD6~AD0**: USB device address bit 6 ~ bit 0

Bit 0 **WKEN**: USB remote wake-up function enable control
0: Disable
1: Enable

STLI/STLO Registers

The STLI/STLO registers show whether the corresponding endpoint has worked properly or not. As soon as an endpoint improper IN/OUT operation occurs, the related bit in the STLI/STLO registers has to be set high by application program. The STLI/STLO registers content will be cleared by a USB reset signal and a setup token event.

- **STLI Register**

Bit	7	6	5	4	3	2	1	0
Name	STLI7	STLI6	STLI5	STLI4	STLI3	STLI2	STLI1	STLI0
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 **STLI7~STLI0**: USB endpoint n FIFO IN operation stall indication
0: Endpoint n FIFO IN operation is not stalled
1: Endpoint n FIFO IN operation is stalled
The STLIn bit is set by user when the USB endpoint n is stalled. The STLIn bit is cleared by a USB reset signal. For endpoint 0 the STLI0 bit can also be cleared by a SETUP token.

- **STLO Register**

Bit	7	6	5	4	3	2	1	0
Name	STLO7	STLO6	STLO5	STLO4	STLO3	STLO2	STLO1	—
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 **STLO7~STLO1**: USB endpoint 7~endpoint 1 FIFO OUT operation stall indication
0: Endpoint n FIFO OUT operation is not stalled
1: Endpoint n FIFO OUT operation is stalled
The STLOn bit is set by user when the USB endpoint n is stalled. The STLOn bit is cleared by a USB reset signal.

Bit 0 Unimplemented, read as "0"

SIES Register

The SIES register is used to indicate the present signal state which the SIE receives and also control whether the SIE changes the device address automatically or not.

Bit	7	6	5	4	3	2	1	0
Name	NMI	CRCF	—	NAK	IN	OUT	ERR	ASET
R/W	R/W	R/W	—	R/W	R	R/W	R/W	R/W
POR	0	0	—	0	0	0	0	0

Bit 7 **NMI**: Endpoint 0 NAK token interrupt mask control
0: Endpoint 0 NAK token interrupt is not masked
1: Endpoint 0 NAK token interrupt is masked
If this bit is set to 1, the interrupt will not be generated when the device sends a NAK token to the USB host from endpoint 0. Otherwise, the endpoint 0 NAK token interrupt will be generated if the corresponding endpoint interrupt control is enabled when this bit is set to 0 and the device endpoint 0 sends a NAK token to the USB host.

Bit 6 **CRCF**: Transfer error indication
0: No USB transfer error occurs
1: USB transfer error occurs
The error conditions include the CRC error, PID error and Bit stuffing error. This bit is set by hardware and cleared by software.

Bit 5 Unimplemented, read as "0"

Bit 4 **NAK**: SIE No-Response indication
0: SIE responds normally
1: SIE no-response
This bit is used to indicate whether the SIE responds normally or not. If there are error conditions detected by the SIE, the SIE will have no response to the USB token. Note that this bit is set by the SIE and cleared by application programs.

Bit 3 **IN**: IN token indication
0: The received token packet is not IN token
1: The received token packet is IN token
The IN bit is used to indicate whether the current token packet received from the USB host is IN token or not.

Bit 2 **OUT**: OUT token indication
0: The received token packet is not OUT token
1: The received token packet is OUT token
The OUT bit is used to indicate whether the token received from the USB host is OUT token or not except the OUT zero length token. This bit should be cleared to 0 by application program after an OUT data has been read. Note that this bit will also be cleared when the next valid SETUP token is received.

Bit 1 **ERR**: FIFO access error indication
0: No error occurs
1: Error occurs
This bit is used to indicate whether the USB bus errors, such as CRC error, PID error or bit stuffing error, etc., has occurred or not when the FIFO is accessed. This bit is set by SIE and cleared by application program.

Bit 0 **ASET**: Device address update control
0: Device address is immediately updated when an address is written into the AWR register
1: Device address is updated after the device IN token data has completely been read by the USB host
This bit is used to configure the SIE to automatically change the device address by the value stored in the AWR register. When this bit is set to "1" by firmware, the SIE will update the device address by the value stored in the AWR register after the USB host

has successfully read the data from the device by an IN operation. Otherwise, when this bit is cleared to "0", the SIE will update the device address immediately after an address is written to the AWR register. Therefore, in order to operate properly, the firmware has to clear this bit after a next valid SETUP token is received.

MISC Register

Bit	7	6	5	4	3	2	1	0
Name	LEN0	READY	SETCMD	D4	E3IDF	CLEAR	TX	REQUEST
R/W	R	R	R/W	R	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **LEN0**: Zero-length packet indication
0: The received packet is not zero-length packet
1: The received packet is zero-length packet
This bit is used to show whether the USB host sends a zero-length packet or not. This bit is set by hardware and cleared by a read operation to the corresponding FIFO.

Bit 6 **READY**: Endpoint FIFO ready indication
0: The desired endpoint FIFO is not ready
1: The desired endpoint FIFO is ready

Bit 5 **SETCMD**: SETUP command indication
0: The data in the endpoint 0 FIFO is not SETUP token
1: The data in the endpoint 0 FIFO is SETUP token
This bit is set by hardware and cleared by application program.

Bit 4 Reserved bit, must be fixed at "0"

Bit 3 **E3IDF**: Endpoint 3 input FIFO size doubling control
0: Single buffer
1: Double buffer

Bit 2 **CLEAR**: FIFO clear function enable control
0: No operation
1: Clear the requested endpoint FIFO
This bit is used to clear the requested FIFO even if the corresponding FIFO is not ready. The CLEAR bit should be set to 1 to generate a positive pulse with a pulse width of 2μs to clear the requested FIFO and then clear this bit to zero.

Bit 1 **TX**: Data transfer direction indication
0: MCU read data from the USB FIFO
1: MCU wirte data to the USB FIFO
This bit defines the data transfer direction between the MCU and USB endpoint FIFO. When the TX bit is set to 1, it means that the MCU wants to write data to the USB endpoint FIFO. After the MCU write operation has completed, this bit has to be cleared to 0 before terminating the FIFO request to indicate the end of the data transfer. For a MCU read operation this bit has to be cleared to 0 to indicate that the MCU wants to read data from the USB endpoint FIFO. Then this bit has to be set to 1 before terminating the FIFO request to indicate the end of the data transfer after an MCU read operation completion.

Bit 0 **REQUEST**: FIFO request control
0: No reqeust or request completion
1: Request desired FIFO
This bit is used to request an operation of the desired endpoint FIFO. After selecting the desired endpoint FIFO, the FIFO can be requested by setting this bit high. Then this bit should be cleared to zero after the operation completion.

PLLC Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1 **PLLF**: PLL clock stable flag
0: PLL clock unstable
1: PLL clock stable
When the PLL bit in the USC register is clear to 0 to enable the PLL circuit, this bit will first be cleared to 0 and then be set to 1 after the PLL clock is stable. The PLL stable time is equal to 2560 PLL clock cycles, 2560 f_{PLL}, after the PLL circuit is enabled.

Bit 0 Unimplemented, read as "0"

UFIEN Register

Bit	7	6	5	4	3	2	1	0
Name	SETI7	SETI6	SETI5	SETI4	SETI3	SETI2	SETI1	FIFO_DEF
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 **SETI7**: Endpoint 7 input FIFO enable control
0: Disable
1: Enable

Bit 6 **SETI6**: Endpoint 6 input FIFO enable control
0: Disable
1: Enable

Bit 5 **SETI5**: Endpoint 5 input FIFO enable control
0: Disable
1: Enable

Bit 4 **SETI4**: Endpoint 4 input FIFO enable control
0: Disable
1: Enable

Bit 3 **SETI3**: Endpoint 3 input FIFO enable control
0: Disable
1: Enable

Bit 2 **SETI2**: Endpoint 2 input FIFO enable control
0: Disable
1: Enable

Bit 1 **SETI1**: Endpoint 1 input FIFO enable control
0: Disable
1: Enable

Bit 0 **FIFO_DEF**: FIFO configuration redefine enable control
0: Disable
1: Enable
If this bit is set to 1, the SIE will redefine the FIFO configuration. Then this bit will be automatically cleared to 0 by the SIE.

UFOEN Register

Bit	7	6	5	4	3	2	1	0
Name	SETO7	SETO6	SETO5	SETO4	SETO3	SETO2	SETO1	DATATG
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 **SETO7**: Endpoint 7 output FIFO enable control
0: Disable
1: Enable

Bit 6 **SETO6**: Endpoint 6 output FIFO enable control
0: Disable
1: Enable

Bit 5 **SETO5**: Endpoint 5 output FIFO enable control
0: Disable
1: Enable

Bit 4 **SETO4**: Endpoint 4 output FIFO enable control
0: Disable, Endpoint 4 input FIFO only

Bit 3 **SETO3**: Endpoint 3 output FIFO enable control
0: Disable
1: Enable

Bit 2 **SETO2**: Endpoint 2 output FIFO enable control
0: Disable
1: Enable

Bit 1 **SETO1**: Endpoint 1 output FIFO enable control
0: Disable
1: Enable

Bit 0 **DATATG**: Data token toggle control
0: DATA0 will be sent first
1: DATA1 will be sent first
This bit is used to select the Data token toggle bit. When this bit is cleared to 0, a DATA0 will first be sent in the following IN or OUT Data pipe for the requested endpoint FIFO. Otherwise, a DATA1 will be sent first followed by the successive IN or OUT data transfer.

UFC0 Register

Bit	7	6	5	4	3	2	1	0
Name	E3FS1	E3FS0	E2FS1	E2FS0	E1FS1	E1FS0	—	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	—
POR	0	0	0	0	0	0	—	—

Bit 7~6 **E3FS1~E3FS0**: Endpoint 3 FIFO size selection
00: 8 bytes
01: 16 bytes
10: 32 bytes
11: 64 bytes

Bit 5~4 **E2FS1~E2FS0**: Endpoint 2 FIFO size selection
00: 8 bytes
01: 16 bytes
10: 32 bytes
11: 64 bytes

Bit 3~2 **E1FS1~E1FS0**: Endpoint 1 FIFO size selection
00: 8 bytes
01: 16 bytes
10: 32 bytes
11: 64 bytes

Bit 1~0 Unimplemented, read as "0"

UFC1 Register

Bit	7	6	5	4	3	2	1	0
Name	E7FS1	E7FS0	E6FS1	E6FS0	E5FS1	E5FS0	E4FS1	E4FS0
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 **E7FS1~E7FS0**: Endpoint 7 FIFO size selection
00: 8 bytes
01: 16 bytes
10: 32 bytes
11: 64 bytes

Bit 5~4 **E6FS1~E6FS0**: Endpoint 6 FIFO size selection
00: 8 bytes
01: 16 bytes
10: 32 bytes
11: 64 bytes

Bit 3~2 **E5FS1~E5FS0**: Endpoint 5 FIFO size selection
00: 8 bytes
01: 16 bytes
10: 32 bytes
11: 64 bytes

Bit 1~0 **E4FS1~E4FS0**: Endpoint 4 FIFO size selection
00: 8 bytes
01: 16 bytes
10: 32 bytes
11: 64 bytes

FIFOn Registers

The FIFOn Register is used for data transactions storages between the USB device and the USB host. The MCU reads data from or writes data to the FIFOn via the specific combination of the corresponding control and selection bits.

Name	Type	POR	Descriptions
FIFO0	R/W	xxxx xxxx	Endpoint 0 Data Pipe
FIFO1	R/W	xxxx xxxx	Endpoint 1 Data Pipe
FIFO2	R/W	xxxx xxxx	Endpoint 2 Data Pipe
FIFO3	R/W	xxxx xxxx	Endpoint 3 Data Pipe
FIFO4	R/W	xxxx xxxx	Endpoint 4 Data Pipe
FIFO5	R/W	xxxx xxxx	Endpoint 5 Data Pipe
FIFO6	R/W	xxxx xxxx	Endpoint 6 Data Pipe
FIFO7	R/W	xxxx xxxx	Endpoint 7 Data Pipe

"x": unknown

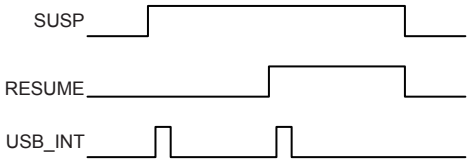
USB Suspend Mode and Wake-up

USB Suspend Mode

If there is no signal on the USB bus for over 3ms, the USB device will enter the suspend mode. The Suspend flag, SUSP, in the USC register will then be set high and an USB interrupt will be generated to indicate that the device should jump to the suspend state to meet the requirements of the USB suspend current specification. In order to meet the requirements of the suspend current; the firmware should disable the USB clock by clearing the USBCKEN bit to "0". The suspend mode current can be further descreased by setting the SUSP2 bit in the UCC register.

USB Host Wake-up

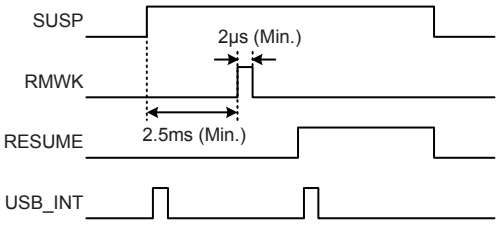
When the resume signal is asserted by the USB host, the device will be woken up by the USB interrupt and the RESUME bit in the USC register will be set. To ensure correct device operation, the application program should set the USBCKEN bit high and the USB host will start to communicate with the USB device. Then the SUSP2 bit will be cleared low together with the RESUME bit when the USB device actually leaves the suspend mode. Therefore, when the device detects the suspend bit, SUSP2, the resume bit, RESUME, should be monitored and taken into consideration.



Suspend and Host Wake-up

USB Remote Wake-up

As the USB device has a remote wake-up function, the USB device can wake up the USB host by sending a remote wake-up pulse which is generated by setting the RMWK bit high and then clearing it low after 2μs later. Once the USB host receives a remote wake-up signal from the USB device, the host will send a resume signal to device.



Suspend and Remote Wake-up

USB Interrupts

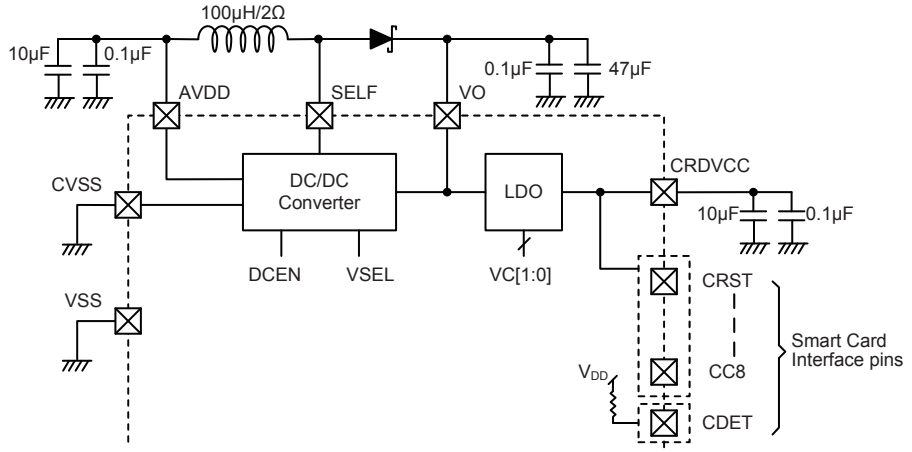
Several USB conditions can generate an USB interrupt. When one of these conditions exists, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are the USB suspended, USB resumed, USB reset and USB endpoint FIFO access events. When the USB interrupt caused by any of these conditions occurs, if the corresponding interrupt control is enabled and the stack is not full, the program will jump to the corresponding interrupt vector where it can be serviced before returning to the main program.

For the USB Endpoint FIFO access event, there are the corresponding indication flags to indicate which endpoint FIFO is accessed. As the Endpoint FIFO access flag is set, it will generate a USB interrupt if the associated Endpoint FIFO pipe and interrupt control are both enabled. The Endpoint FIFO access flags should be cleared by the application program. As the USB suspended or USB resume condition occurs, the corresponding indication flag, known as SUSP and RESUME bits, will be set and a USB interrupt will directly generate without enabling the associated interrupt control bit. The SUSP and RESUME bits are read only and set or cleared by the USB SIE. For a USB interrupt occurred to be serviced, in addition to the bits for the corresponding interrupt enable control in USB module being set, the global interrupt enable control and the related interrupt enable control bits in the host MCU must also be set. If these bits are not set, then no interrupt will be serviced.

DC/DC Converter and LDO

The series devices contains a DC/DC Converter and an LDO to provide the power supply for the Smart Card interface pins and the external Smart Card. The DC/DC Converter is a PFM step-up DC/DC converter with high efficiency and low ripple. It requires only three external components to provide an output voltage of either 3.8V or 5.5V selected by the DC/DC output voltage selection bit VSELin the DC2DC register. The DC/DC voltage output is connected to an external pin as well as being internally connected to the LDO input. It also contains an enable control bit, DCEN, in the DC2DC register to reduce power consumption when in the power down mode. If the Smart Card voltage output is switched to 0V by clearing the selection bits VC[1:0] in the CCR register, the DC/DC converter will automatically be turned off even if the enable control bit DCEN is set to 1.

The LDO is a three-terminal high current low voltage dropout regulator with over current protection. It supports three output voltages of 1.8V, 3.0V or 5.0V selected by the Smart Card Voltage selection bits VC1 and VC0 in the CCR register. It can deliver a minimum output current of 35mA, 55mA and 55mA when the LDO output voltage is 1.8V, 3.0V and 5.0V respectively.



DC/DC Converter and LDO Diagram

DC2DC Register

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

Bit 7~2 Unimplemented, read as "0"

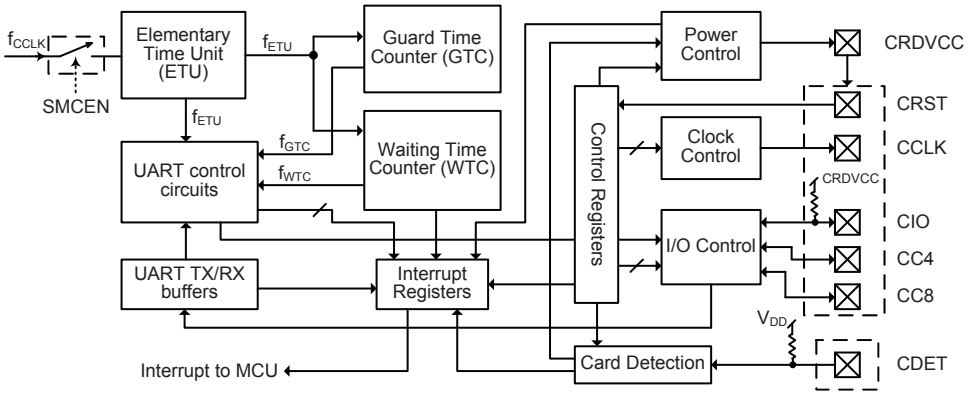
Bit 1 **DCEN**: DC/DC converter enable control
0: Disable
1: Enable
This bit is used to control the DC/DC converter function. If this bit is set to 1, the DC/DC output voltage is selected by the VSEL selection bit. If this bit is cleared to 0, the DC/DC converter function is disabled and the output voltage is equal to the value of (V_{DD}–V_{DIODE}).

Bit 0 **VSEL**: DC/DC converter output voltage selection
0: 3.8V
1: 5.5V

Smart Card Interface

The series devices contains a Smart Card Interface compatible with the ISO 7816-3 standard. This interface includes the Card Insertion/Removal detection, UART interface control logic and data buffers, Power control circuits, internal Timer Counters and control logic circuits to perform the required Smart Card operations. The Smart Card interface acts as a Smart Card Reader to facilitate communication with the external Smart Card. The overall functions of the Smart Card interface is control by a series of registers including control and status registers.

As the complexity of ISO7816-3 standard data protocol does not permit comprehensive specifications to be provided in this datasheet, the reader should therefore consult other external information for a detailed understanding of this standard.



Interface Pins

To communicate with an external Smart Card, the internal Smart Card interface has a series of external pins known as CRDVCC, CRST, CCLK, CIO, CC4, CC8 and CDET. The CRDVCC pin is the power supply pin of the external Smart Card and the Smart Card interface pins described above except the CDET pin. It can output several voltage levels as selected by the VC1 and VC0 selection bits. The CRST pin is the reset output signal which is used to reset the external Smart Card. Together with the internal CRST control bit the MCU can control the CRST pin level by writing specific data to the CRST bit and obtain the CRST pin status by reading the CRST bit. The CCLK pin is the clock output signal used to communicate with the external Smart Card together with the serial data pin, CIO. The operation of CCLK and CIO can be selected as the UART mode automatically driven by the UART control circuits, or the Manual mode controlled by configuring the internal CCLK and CIO bits respectively by the application program. The CDET pin is the external Smart Card detection input pin. When the external Smart Card is inserted or removed, it can be detected and generate an interrupt signal which is sent to MCU if the corresponding interrupt control bit is enabled. The CC4 and CC8 pins are used as I/O pins and are controlled by the corresponding CC4 and CC8 bits.

Card Detection

If an external Smart Card is inserted, the internal card detector can detect this insertion operation and generate a Card insertion interrupt. When the Card is present and detected, the power-on sequence for the external Smart Card should be activated by the application programs to supply power for the external Smart Card. Similarly, if the Card is removed, the internal card detector can also detect the removal and consequently generate a Card removal interrupt. Like the Card insertion operation, the Card Removal deactivation procedure defined in the ISO 7816-3 standard should be activated by the application programs.

The card detector can support two kinds of card detect switch mechanisms. One is a normally open switch mechanism when the card is not present and the other is a normally closed switch mechanism. After noting which card detect switch mechanism type is used, the card switch selection should be configured by setting the selection bit CDET in the CCR Register to correctly detect the Card presence. No matter what type of the card switch is selected by configuring the CDET bit, the Card Insertion/Removal Flag, CIRF, in the CSR register will be set to ″1″ when the card is actually present on the CDET pin and shoule be clear to “0” by the application program. Note that there is no hardware debounce circuits in the card detector. Any change of the CDET pin level will cause the CIRF bit to change. The required debounce time should be handled by the application program.

There is a pull-high resistor integrated in the card detector. A CDET pin pull-high resistor enable control bit, RCDET, in the ISOC register can determine whether the pull high resistor is internally connected to the CDET pin or not.

Internal Time Counter – ETU, GTC, WTC

For proper data transfer, some timing purposed setting procedures must be executed before the Smart Card Interface can begin to communicate with the external card. There are three internal counters named Elementary Time Unit, ETU, Guard Time Counter, GTC, and Waiting Time Counter, WTC which are used for the timing related functions in the Smart Card interface operation.

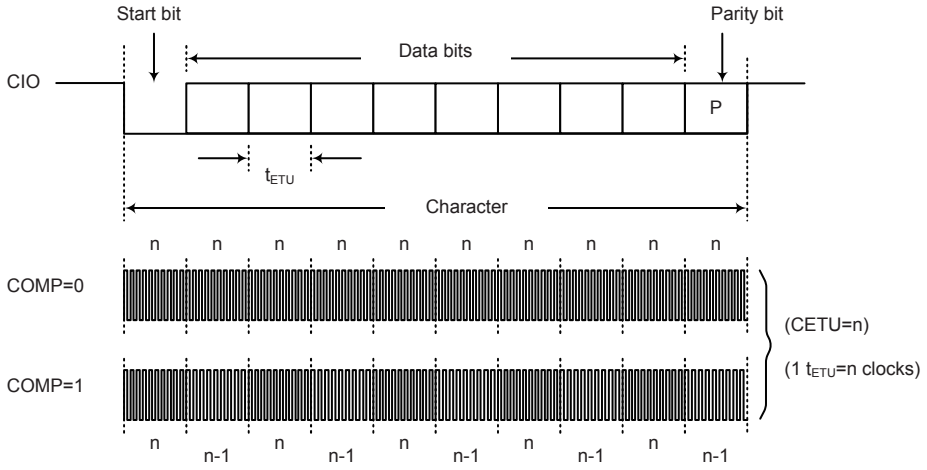
Elementary Time Unit – ETU

The Elementary Time Unit, ETU, is an 11-bit up-counting counter and generates the clock, denoted as f_{ETU} to be used as the operating frequency for the UART transmission and reception in the Smart Card interface. The clock source of the ETU named as f_{CCLK} comes from the f_{CRD} clock and the frequency of f_{CCLK} can be f_{CRD} or $f_{CRD}/2$ which is selected using the SMF bit in the ISOC register. The f_{CRD} clock is derived from the high speed clock, f_H, and the f_{CRD} frequency can be equal to the frequency of f_H, $f_H/2$, $f_H/3$ or $f_H/4$ selected by the Smart Card clock source selection bits, CRDCKS1 and CRDCKS0. The data transfer of the UART interface is a character frame based protocol, which basically consists of a Start bit, 8-bit data and a Parity bit. The time period t_{ETU} ($1/f_{ETU}$), generated by the ETU, is the time unit for UART character bit. There are two registers related to the ETU known as the low byte ETU register CETU0 and the high byte ETU register CETU1, which store the expected contents of the ETU. Each time the high byte ETU register CETU1 is written, the ETU will reload the new written value and restart counting. The elementary time unit tETU is obtained from the following formula.

$$t_{ETU} = \frac{F}{D} \times \frac{1}{f},$$

F: clock rate conversion integer
D: baud rate adjustment integer
f : clock rate of Smart Card

The values of F and D, as they appear in the above formula, will be obtained from the Answer to-Reset packet sent from the external Smart Card to the Smart Card interface, the first time the external Smart Card is inserted. When the Smart Card interface receives this information, the values which should be written into the CETU0 and CETU1 can be calculated by F/D. As the value of the ETU registers is obtained by the above formula, the calculation results of the value may not be an integer. If the calculation result is not an integer and is less than the integer n but greater than the integer (n-1), either the integer n or (n-1) should be written into the CETU0 and CETU1 registers depending upon whether the result is closer to n or (n-1). The integer n mentioned here is a decimal. If the calculation result is close to the value of (n-0.5), the compensation mode should be enabled by setting the compensation enable control bit COMP in the CETU1 register to 1 for successful data transfer. When the result is close to the value of (n-0.5) and the compensation mode is enabled, the value written into the CETU0 and CETU1 registers should be n and then the ETU will generate the time unit sequence with n clock cycles and next (n-1) clock cycles alternately and so on. This results in an average time unit of (n-0.5) clock cycles and allows the time granularity down to a half clock cycle. Note that the ETU will reload the ETU registers value and restart counting at the time when the Start bit appears in the UART mode.



Character Frame and Compensation Mode

Guard Time Counter – GTC

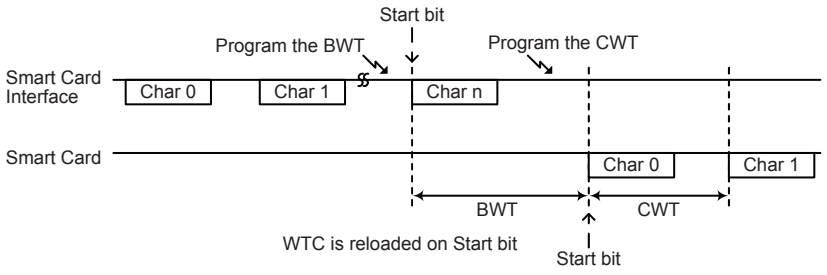
The Guard Time Counter, GTC, is a 9-bit up-counting counter and generates the minimum time duration known as a character frame denoted as t_{GTC} between two successive characters in a UART transmission. The clock source of the GTC comes from the ETU output clock named f_{ETU}. The character transmission rate of the UART interface is controlled by t_{GTC} generated by the GTC. There are two registers related to the GTC known as the low byte GTC register, CGT0, and the high byte GTC register, CGT1, which store the expected GTC value. The GTC value will be reloaded at the end of the current guard time period. Note that the guard time between the last character received from the Smart Card and the next character transmitted by the Smart Card interface (Smart Card reader) should be managed by the application program.

Waiting Time Counter – WTC

The Waiting Time Counter, WTC, is a 24-bit down-counting counter and generates a maximum time duration denoted as t_{WTC} for the data transmission. The WTC clock source comes from the ETU output clock named f_{ETU}. The data transfer is categorized into 2 types. One is the Character transfer which means each data transmission or reception is one character. The other is the Blocks transfer which means each data transmission or reception is more than one character. The information related to the data transfer type or the number of the characters to be transferred is contained in the Answer-to-Reset packet.

There are three data registers for the WTC known as the low byte WTC register CWT0, the middle byte WTC register CWT1 and the high byte WTC register CWT2, which are used to store the expected WTC values. The WTC can be used in both UART mode and Manual mode and can reload the value at specific conditions. The WTC function is controlled by the WTEN bit in the CCR register. When the UART interface is set to be operated in the UART mode and the WTC is enabled, the updated CWT value will be loaded into the WTC as the Start bit is detected. If the UART interface is set to be operated in the Manual mode and the WTC is enabled, the updated CWT value will immediately be loaded into the WTC once the CWTn registers are written into with a new vlaue. Regardless of whether it is in the UART mode or Manual mode, if the WTEN bit is cleared to ″0″, the updated CWT value will not be loaded into the WTC until the WTEN bit is again set to 1 and the WTC underflows from the current loaded value.

When the transfer type is configured as a Character transfer, the WTC will generate the maximum timeout period of the Character Waiting Time, CWT. If the transfer type is configured as a Block transfer, the WTC will generate the Character Waiting Time, CWT, except for the last character. The Block Waiting Time, BWT, should be loaded into the WTC data registers before the Start bit of the last transmitted character occurs. As the Start bit of the last transmitted character occurs, the BWT value will be load into the WTC. Then the Smart Card may be expected to transmit data to the Smart Card interface in the BWT duration. If the Smart Card does not transmit any data characters, the WTC will underflow. When the WTC underflows, the corresponding request flag WTF in the CSR register will be set. The Waiting Time Underflow pending flag WTP in the CIPR register will also be set if the interrupt enable control bit WTE in the CIER register is set. Then an interrupt will be generated to notify the MCU that the Smart Card has not responded to the Smart Card reader. Note that if the WTC value is set to zero, the WTF bit will be equal to ″1″.



Smart Card UART Mode

Data transfer with the external Smart Card is implemented into two operating modes. One is the UART mode while the other is the Manual mode. The data transfer mode is selected by the UART mode selection bit, UMOD, in the CCR register. When the UMOD bit is set to 1, the UART mode is enabled and data transfer operates in the UART mode. Otherwise, data transfer operates in the Manual mode if the UMOD bit is set to 0. The UART interface is a half-duplex interface and communicates with the external Smart Card via the CCLK and CIO pins. The CIO pin can be selected to be connected to a pull high resistor by the RCIO bit in the ISOC register. After a reset condition the UART function is in the reception mode but the UART mode is disabled. When the UART mode is selected, data transfer is driven by the UART circuits automatically through the CCLK and CIO pins.

There are two data registers related to data transmission and reception, CTXB and CRXB, which store the data to be transmitted and received respectively. If a character is written into the CTXB register in the UART mode, the UART interface will automatically switch to the transmission mode from the reception mode after a reset. When the UART transmission or reception has finished, the corresponding request flag named TXCF or RXCF is set to 1. If the transmit buffer is empty, the transmit buffer empty flag, TXBEF, will be set to 1.

The UART interface supports a parity generator and a parity check function. As the parity error occurs during a data transfer, the corresponding request flag named, PARF, in the CSR register will be set to 1. Once the PARF bit is set to 1, the Parity error pending flag PARP in the CIPR register will be set to 1 if the relevant interrupt control bit is enabled. Then an interrupt signal will be generated and sent to the MCU.

There is a Character Repetition function supported by the UART interface when a parity error occurs. The Character repetition function is enabled by setting the CREP bit to 1 and then the repetition function is activated when the parity error occurs during data transfers. The repetition times can be selected to be 4 or 5 times by the RETRY45 bit in the ISOC register. When the CREP bit is set to 1 and the repetition times is set to 4 times, the UART interface, if in the transmission mode, will transmit the data repeatedly for 4 times at most when an error signal occurs. If the data transmitted by the UART interface is received by the Smart Card receiver without a parity error during these 4 transmissions, the transmission request flag, TXCF, of the UART interface will be set to 1 and the PARF bit will be cleared to 0. If the UART interface is informed that there is still an error signal during the 4 transmissions, the parity error flag PARF of the UART interface will be set to 1 at the end of the 4th transmission but the transmission request flag TXCF will not be set. If the UART interface is in the reception mode together with the CREP bit being set to 1, it will inform the external Smart Card transmitter that there is a parity error for at most 4 times. If the data transmitted by the external Smart Card transmitter is received by the UART interface without a parity error during the 4 receptions, the reception request flag, RXCF, of the UART interface will be set to 1 and the PARF bit will be cleared to 0. If there is still a parity error during the 4 data receptions, the UART interface will inform the external Smart Card and the parity error flag, PARF, of the UART interface will be set to 1 at the end of the 4th reception as well as the reception request flag, RXCF.

If the CREP bit is set to 0 and the UART interface is in the reception mode, both the PARF and RXCF bits will be set to 1 when the data with parity error has been received but the character repetition will not be activated. If the UART interface is in the transmission mode and the CREP bit is set to 0, it acts as a normal transmitter and the TXCF bit is set to 1 after the data has been transmitted. It has no effect on PARF bit.

When data is selected to be transferred in the Manual mode by setting the UMOD bit to 0, the data is controlled by the control bit, CIO, in the CCCR register. The value of the CIO bit will be reflected immediately on the CIO pin in the Manual mode. Note that in the Manual mode the character repetition function is not available as well as the related flags and all the data transfer is handled by the application program. The clock used to drive the external Smart Card that appears on the CCLK pin can be the f_{CCLK} clock which is derived from the internal clock source named as the f_{CRD} clock or the control bit CCLK in CCCR register and selected by the Smart Card selection bit CLKSEL in the CCCR register. When CLKSEL is set to 1 to select the f_{CCLK} clock as the Smart Card clock source, a software control bit, SMF, can determine whether the clock output on the CCLK pin, which comes from the f_{CRD} clock, is moreover to be divided by 2 or not. If users wish to handle the CCLK clock manually, the CLKSEL bit should first be set to 0 and then the value of the CCLK bit will be present on the CCLK pin.

When the Smart Card is first inserted, the data direction convention is sent first in the Answer-to Reset packet to inform the Smart Card interface whether the MSB of the data is sent first or the LSB is sent first. If the direction convention used by the Smart Card is the same as the convention used by the Smart Card interface, the UART interface will generate a reception interrupt if the reception interrupt is enabled without a parity error flag. Otherwise, the UART interface will generate a reception interrupt and the parity error flag will be asserted. By checking the parity error flag the Smart Card interface can know if the data direction convention is correct or not.

Power Control

When the Smart Card is first inserted and detected, the power-on sequence for the external Smart Card should be activated by the application programs to supply power to the external Smart Card. All the information necessary for the Smart Card interface to communicate with the external Card is contained in the Answer-to-Reset packet including the data transfer type (Character or Blocks), the data direction convention (MSB or LSB first), the clock rate information (ETU, GTC or WTC), etc. The voltage level supplied to the Smart Card is also defined in the Answer-to-Reset packet. The Smart Card power supply voltage level is generated by the LDO and selected by the Smart Card Power Supply voltage selection bits VC1 and VC0 in the CCR register. When the external Smart Card is inserted, the application program should set the CRDVCC pin to the expected voltage level defined in the Answer-to-Reset packet. Similarly, the power deactivation procedure defined in the ISO 7816-3 standard should also be properly arranged by the application programs when the external Smart Card is removed. After the external Smart Card is removed, the Smart Card Interface Circuitry enable control bit, CVCC, in the CCCR Register will be automatically cleared to zero to prevent the Smart Card Interface module from being re-modified.The CRDVCC pin voltage level and the related Smart Card interface register contents will remain unchanged but the relevant Smart Card Interface pins including CRST, CCLK, CIO, CC4 and CC8 pins will be kept at a low level when the external Card is removed.

The Power Control circuitry provides the Card voltage and current indicators to avoid malfunctions. When the Card voltage is within its specified range, the Card Voltage flag VCOK will be set to 1. If the Card is not in the specified range, the VCOK flag will be cleared to 0 to indicate that the Card voltage is not within the specified range. As the VCOK bit changes from 1 to 0, the corresponding pending flag named VCP in CIPR register will be set to 1 if the Card Voltage error interrupt control VCE in the CIER register is enabled.

When the current consumed by the external Smart Card is within the range specified in the ISO 7816-3 standard, the Card Current Overload flag IOVF will remain at a "0" value. If the Card current is not within the specified range, the IOVF flag will be set to 1 to indicate that the Card Current overloads. As the IOVF bit is set to 1, the relevant pending flag, IOVFP, in the CIPR register will also be set to 1 if the Card Current Overload interrupt control enable bit, IOVFE, in the CIER register is enabled.

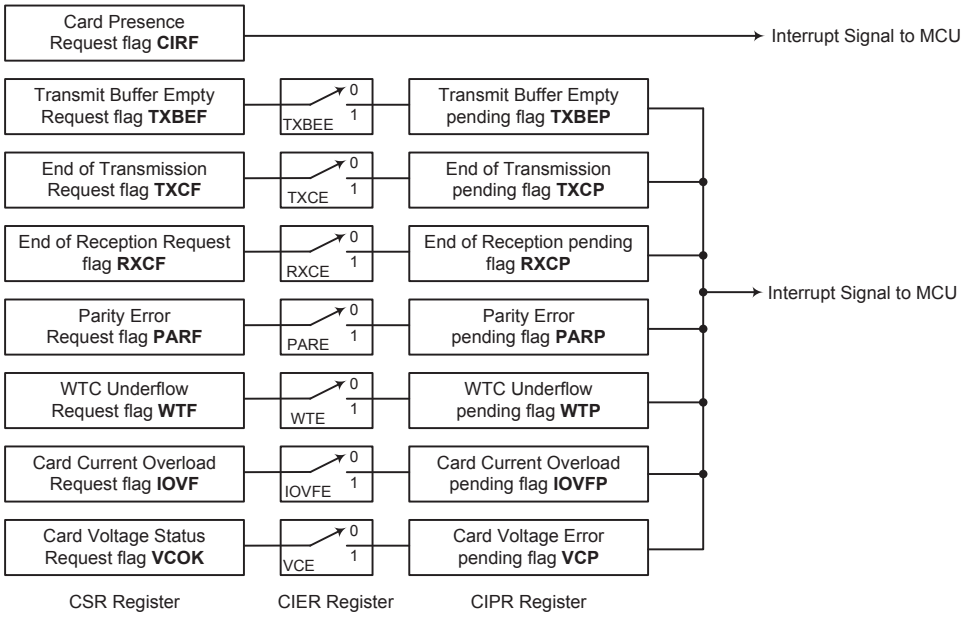
Smart Card Interrupt Structure

There are several conditions for the Smart Card that to generate a Smart Card interrupt. When these conditions exist, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a Smart Card Insertion/Removal, a Smart Card Voltage error, a Smart Card Current Overload, a Waiting Time Counter Underflow, a Parity error, an end of a Character Transmission or Reception and an empty Transmit buffer. When a Smart Card interrupt is generated by any of these conditions, then if the corresponding interrupt enable control bit in the host MCU is enabled and the stack is not full, the program will jump to the corresponding interrupt vector where it can be serviced before returning to the main program.

For Smart Card interrupt events, except for Card Insertion/Removal events, there are corresponding pending flags which can be masked by the corresponding interrupt enable control bits. When the related interrupt enable control is disabled, the corresponding interrupt pending flag will not be affected by the request flag and no interrupt will be generated. If the related interrupt enable control is enabled, the relevant interrupt pending flag will be affected by the request flag and then the interrupt will be generated. The pending flag register CIPR is read only and once the pending flag is read by the application program, it will be automatically cleared while the related request flag should be cleared by the application program manually.

When a Smart Card Insertion/Removal event occurs, the Card Insertion/Removal request flag, CIRF, will be set or clear dependent upon the presence of a card, and a Smart Card Insertion/Removal interrupt will be directly generated without any associated Smart Card interrupt control being enabled.

For a Smart Card interrupt occurred to be serviced, in addition to the bits for the corresponding interrupt enable control in the Smart Card interface being set, the global interrupt enable control and the related interrupt enable control bits in the host MCU must also be set. If these bits are not set, then no interrupt will be serviced.



Smart Card Interrupt Structure

Programming Considerations

Since the whole Smart Card interface is driven by the clock f_{CRD} which is derived from the high speed oscillator clock f_H, the Smart Card interface will not operate, even interface registers read/write operations, if the high speed oscillator clock f_H is stopped. For example, if the MCU clock source is switched to the low speed clock f_{SUB} which comes from the low speed oscillator LXT or LIRC, then all operations related to the Smart Card interface are not performed.

Smart Card Interface Status and Control Registers

There are several registers associated with the Smart Card function. Some of the registers control the overall function of the Smart Card interface as well as the interrupts, while some of the registers contain the status bits which indicate the Smart Card data transfer situations, error conditions and power supply conditions. Also there are two registers for the UART transmission and reception respectively to store the data received from or to be transmitted to the external Smart Card.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CCR	RSTCRD	CDET	VC1	VC0	UMOD	WTEN	CREP	CONV
CSR	TXBEF	CIRF	IOVF	VCOK	WTF	TXCF	RXCF	PARF
CCCR	CLKSEL	—	CC8	CC4	CIO	CCLK	CRST	CVCC
ISOC	CRDCKS1	CRDCKS0	SMF	SMCEN	—	RETRY45	RCIO	RCDET
CETU1	COMP	—	—	—	—	ETU10	ETU9	ETU8
CETU0	ETU7	ETU6	ETU5	ETU4	ETU3	ETU2	ETU1	ETU0
CGT1	—	—	—	—	—	—	—	GT8
CGT0	GT7	GT6	GT5	GT4	GT3	GT2	GT1	GT0
CWT2	WT23	WT22	WT21	WT20	WT19	WT18	WT17	WT16
CWT1	WT15	WT14	WT13	WT12	WT11	WT10	WT9	WT8
CWT0	WT7	WT6	WT5	WT4	WT3	WT2	WT1	WT0
CIER	TXBEE	—	IOVFE	VCE	WTE	TXCE	RXCE	PARE
CIPR	TXBEP	—	IOVFP	VCP	WTP	TXCP	RXCP	PARP
CTXB	TB7	TB6	TB5	TB4	TB3	TB2	TB1	TB0
CRXB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0

Smart Card Interface Registers List

CCR Register

Bit	7	6	5	4	3	2	1	0
Name	RSTCRD	CDET	VC1	VC0	UMOD	WTEN	CREP	CONV
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 **RSTCRD**: Smart card interface UART reset
0: No Smart card UART reset
1: Smart card UART reset
This bit is used to reset the smart card UART by setting this bit high. This bit is set and cleared by application program.

Bit 6 **CDET**: Card presence detector configuration
0: Connector switch open if no card is present
1: Connector switch short if no card is present
This bit is set and cleared by application program to configure the switch type of the card presence detector.

Bit 5~4 **VC1~VC0**: Card voltage selection
00: 0V
01: 1.8V
10: 3V
11: 5V
These bits are set and cleared by application program to select the card voltage supplied on the CRDVCC pin.

Bit 3 **UMOD**: UART mode selection
0: Manual mode – CIO is driven by application program via CIO bit
1: UART mdoe – CIO is controlled by the smart card UART module
This bit is set and cleared by application program to select the smart card UART operating mode.

Bit 2 **WTEN**: Waiting Time Counter (WTC) counting control
0: WTC stops counting
1: WTC starts to count
This bit is set and cleared by application program to enable the WTC counting function. When the WTEN bit is cleared to 0, a write access to the CWT2 register will load the value held in the CWT2~CWT0 registers into the WTC. If it is set to 1, the WTC is enabled and automatically reloaded with the value in CWT2~CWT0 at each start bit occurrence.

Bit 1 **CREP**: Character repetition enable control at a parity error condition
0: No retry on parity error
1: Automatically retry on parity error
The CREP bit is set and cleared by application program. When the CREP bit is cleared to 0, both the RXCF and PARF flags will be set on parity error in reception mode after the data is received while the PARF is set but the TXCF is cleared in the transmission mode. If the CREP bit is set to 1, the character retry will automatically be activated on parity error for 4 or 5 times depending upon the RETRY45 bit in the ISOC register. In the transmission mode the character will be re-transmitted if the transmitted data is refused and then the parity error flag PARF will be set at the end of the 4th or 5th transmission but the TXCF bit will not be set. In the reception mode if the received data has a parity error, the receiver will inform the transmitter for 4 or 5 times and then the PARF and RXCF flags will both be set at the end of the 4th or 5th reception.

Bit 0 **CONV**: Data direction convention
0: LSB is transferred first – State “High” encodes value “1” and LSB is transferred first
0: MSB is transferred first – State “Low” encodes value “1” and MSB is transferred first
This bit is set and cleared by application program to select if the data is LSB transferred first or MSB transferred first. When the data direction is the same as the direction specified by the external Smart Card, the RXCF bit will be set to 1 without a parity error. Otherwise, both the RXCF and PARF bits will be set to 1 after the data is received.

CSR Register

Bit	7	6	5	4	3	2	1	0
Name	TXBEF	CIRF	IOVF	VCOK	WTF	TXCF	RXCF	PARF
R/W	R	R	R	R	R	R/W	R	R/W
POR	1	0	0	0	0	0	0	0

Bit 7 **TXBEF**: Transmit buffer empty request flag
0: Transmit buffer is not empty
1: Transmit buffer is empty
This bit is used to indicate whether the transmit buffer is empty or not and is set or cleared by hardware automatically.

Bit 6 **CIRF**: Card presence request flag
0: No card is present
1: A card is present
This bit is used to indicate whether the card is present or not and is set or cleared by hardware automatically. The CIRF bit will synchronously trigger the SCIRF bit in the MCU interrupt controller.

Bit 5 **IOVF**: Card current overload request flag
0: Card current does not overload
1: Card current overloads
This bit is used to indicate whether the card current overloads or not and is set or cleared by hardware automatically.

Bit 4 **VCOK**: Card voltage status flag
0: Card voltage is not in the specified range
1: Card voltage is in the specified range
This bit is used to indicate whether the card voltage is in the specified range or not and is set or cleared by hardware automatically.

Bit 3 **WTF**: Waiting Time Counter (WTC) underflow request flag
0: WTC does not underflow
1: WTC underflows
This bit is used to indicate whether the WTC underflow or not. It is set by hardware and cleared by the application program in the specific procedure: clr WTEN, access CWT2 and set WTEN.

Bit 2 **TXCF**: Character transmission request flag
0: No character is transmitted
1: A character has been transmitted
The TXCF bit is set by hardware and cleared by the application program. It is used to indicate whether the character has been transmitted or not.

Bit 1 **RXCF**: Character reception request flag
0: No character is received
1: A character has been received
The RXCF bit is set by hardware automatically and cleared after a read access to the CRXB register by the application program. The RXCF bit will always be set to 1 when a character is received regardless of the parity check result. When the character has been received, the received data stored in the CRXB register should be moved to the data memory specified by users. If the contents of the CRXB register are not read before the end of the next character shifted in, the data stored in the CRXB register will be overwritten.

Bit 0 **PARF**: Parity error request flag
0: No parity error
1: Parity error has occurred

CCCR Register

Bit	7	6	5	4	3	2	1	0
Name	CLKSEL	—	CC8	CC4	CIO	CCLK	CRST	CVCC
R/W	R/W	—	R/W	R/W	R	R/W	R/W	R/W
POR	0	—	x	x	x	0	x	0

"x": unknown

Bit 7 **CLKSEL**: Smart card CCLK pin clock source selection
0: From the CCLK bit value
1: From f_{CCLK} clock
This bit is used to select which clock source is present on the external CCLK pin. It is set and cleared by application program. It is recommended that a certain value should be written into the CCLK bit to activate the clock at a known level before the CLKSEL bit is switched from 1 to 0. For detailed f_{CCLK} selection, refer to the ISOC register.

Bit 6 Unimplemented, read as "0"

Bit 5 **CC8**: CC8 pin control
0: The external CC8 pin status is 0
1: The external CC8 pin status is 1
This bit is set and cleared by the application program to control the external CC8 pin status. The value written into this bit will be present on the external CC8 pin. Reading this bit will return the status present on the CC8 pin.

Bit 4 **CC4**: CC4 pin control
0: The external CC4 pin status is 0
1: The external CC4 pin status is 1
This bit is set and cleared by the application program to control the external CC4 pin status. The value written into this bit will be present on the external CC4 pin. Reading this bit will return the status present on the CC4 pin.

Bit 3 **CIO**: CIO pin control
0: The external CIO pin status is 0
1: The external CIO pin has an open drain condition
This bit is only available if the UMOD bit in the CCR register is cleared to 0 to select the Manual mode. It is set and cleared by the application program to control the external CIO pin status in the Manual mode. Reading this bit will return the status present on the CIO pin. A pull high resistor can be connected to the CIO pin determined by the RCIO bit in the ISOC register.

Bit 2 **CCLK**: CCLK pin control
0: The external CCLK pin status is 0
1: The external CCLK pin status is 1
This bit is only available if the CLKSEL bit in the CCCR register is set to 0. It is set and cleared by the application program to control the external CCLK pin status when the CLKSEL bit is low. Reading this bit will return the current value in this bit instead of the status present on the CCLK pin.

Bit 1 **CRST**: CRST pin control
0: The status of the external CRST pin is 0
1: The status of the external CRST pin is 1
This bit is set and cleared by application program to control the external CRST pin status to be used to reset the external Smart Card. Reading this bit will return the present status of the CRST pin.

Bit 0 **CVCC**: Smart Card Interface Circuitry enable control
0: Smart card interface circuitry is disabled.
1: Smart card interface circuitry is enabled.
This bit is set and cleared by application program to control the Smart card interface module is switched on or off when the external Card is present. If the external Card is not present on the CDET pin, this bit is not available to control the Smart card interface circuitry and will automatically be cleared to 0 by hardware.

ISOC Register

Bit	7	6	5	4	3	2	1	0
Name	CRDCKS1	CRDCKS0	SMF	SMCEN	—	RETRY45	RCIO	RCDET
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 7~6 **CRDCKS1~CRDCKS0**: Smart card interface clock source f_{CRD} divided ratio selection
00: $f_{CRD} = f_H /2$
01: $f_{CRD} = f_H /3$
10: $f_{CRD} = f_H /1$
11: $f_{CRD} = f_H /4$

Bit 5 **SMF**: Smart card interface clock frequency f_{CCLK} selection
0: $f_{CCLK} = f_{CRD}$
1: $f_{CCLK} = f_{CRD} /2$

Bit 4 **SMCEN**: Smart card interface clock enable control
0: f_{CCLK} is disabled
1: f_{CCLK} is enabled
This bit is used to control the Smart Card interface clock. If this bit is cleared to disable the clock, the relevant registers in the Smart Card interface module can not be accessed. When the Smart Card interface clock is disabled, it has no effect on the Card insertion or removal detections.

Bit 3 Unimplemented, read as "0"

Bit 2 **RETRY45**: Card interface character transfer repetition times selection
0: 4 times
1: 5 times

Bit 1 **RCIO**: Card interface CIO pin pull high function enable control
0: Disable
1: Enable

Bit 0 **RCDET**: Card interface CDET pin pull high function enable control
0: Disable
1: Enable

CETU Registers

The CETU registers, CETU1 and CETU0, contains the specific value determined by the formula described in the ETU section. It also includes a control bit of the Compensation function for the ETU time granularity. Note that the value of the ETU must be in the range of 001H to 7FFH. To obtain the maximum ETU decimal value of 2048, a "000H" value should be written into the ETU10 ~ ETU0 bits.

- **CETU1 Register**

Bit	7	6	5	4	3	2	1	0
Name	COMP	—	—	—	—	ETU10	ETU9	ETU8
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	1

Bit 7 **COMP**: Compensation function enable control
0: Disable
1: Enable
This bit is set and cleared by application program used to control the compensation function. The compensation function has been described and more details can be obtained in the ETU section.

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **ETU10~ETU8**: ETU bit 10 ~ bit 8 values
The bits are set and cleared by application program to modify the ETU values. Writing to the CETU1 register will reload the updated value into the ETU counter.

- **CETU0 Register**

Bit	7	6	5	4	3	2	1	0
Name	ETU7	ETU6	ETU5	ETU4	ETU3	ETU2	ETU1	ETU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	1	1	0	1	0	0

Bit 7~0 **ETU7~ETU0**: ETU bit 7 ~ bit 0 values
The bits are set and cleared by application program to modify the ETU values.

CGT Registers

The CGT registers, CGT1 and CGT0, store the specific GTC value obtained from the Answer-to-Reset packet described in the GTC section. The GT8~GT0 bits are set and cleared by application program to modify the GTC values. The updated GTC value will be loaded into the GTC counter at the end of the current guard time period. Note that the GTC value must be in the range of 00CH to 1FFH.

- **CGT1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	GT8
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **GT8**: GTC value bit 8

- **CGT0 Register**

Bit	7	6	5	4	3	2	1	0
Name	GT7	GT6	GT5	GT4	GT3	GT2	GT1	GT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	1	0	0

Bit 7~0 **GT7~GT0**: GTC value bit 7 ~ bit 0

CWT Registers

The CWT registers, CWT2~CWT0, store the specific WTC value obtained from the Answer-to-Reset packet described in the WTC section. The WT23~WT0 bits are set and cleared by application program to modify the WTC values. The reload conditions of the updated WTC values are described in the WTC section. Note that the WTC value must be in the range of 0x00-2580H to 0xFF-FFFFH. Refer to the WTC section for more details.

- **CWT2 Register**

Bit	7	6	5	4	3	2	1	0
Name	WT23	WT22	WT21	WT20	WT19	WT18	WT17	WT16
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 **WT23~WT16**: WTC value bit 23 ~ bit 16

- **CWT1 Register**

Bit	7	6	5	4	3	2	1	0
Name	WT15	WT14	WT13	WT12	WT11	WT10	WT9	WT8
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 **WT15~WT8**: WTC value bit 15 ~ bit 8

- **CWT0 Register**

Bit	7	6	5	4	3	2	1	0
Name	WT7	WT6	WT5	WT4	WT3	WT2	WT1	WT0
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 **WT7~WT0**: WTC value bit 7 ~ bit 0

CIER Register

Bit	7	6	5	4	3	2	1	0
Name	TXBEE	—	IOVFE	VCE	WTE	TXCE	RXCE	PARE
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 **TXBEE**: Transmit buffer empty interrupt control
0: Disable
1: Enable
This bit is set and cleared by application program used to control the transmit buffer empty interrupt. If this bit is set to 1, the transmit buffer empty interrupt will be generated when the transmit buffer is empty.

Bit 6 Unimplemented, read as "0"

Bit 5 **IOVFE**: Card current overload interrupt control
0: Disable
1: Enable
This bit is set and cleared by application program and is used to control the card current overload interrupt. If this bit is set to 1, the card current overload interrupt will be generated when the card current overloads.

Bit 4 **VCE**: Card voltage error interrupt control
0: Disable
1: Enable
This bit is set and cleared by application program and is used to control the card voltage error interrupt. If this bit is set to 1, the card voltage error interrupt will be generated when the card voltage is not in the specified ramge.

Bit 3 **WTE**: Waiting time counter underflow interrupt control
0: Disable
1: Enable
This bit is set and cleared by application program and is used to control the waiting time counter underflow interrupt. If this bit is set to 1, the waiting time counter underflow interrupt will be generated when the waiting time counter underflows.

Bit 2 **TXCE**: Character transmission completion interrupt control
0: Disable
1: Enable
This bit is set and cleared by application program and is used to control the character transmission completion interrupt. If this bit is set to 1, the character transmission completion interrupt will be generated at the end of the character transmission.

Bit 1 **RXCE**: Character reception completion interrupt control
0: Disable
1: Enable
This bit is set and cleared by application program and is used to control the character reception completion interrupt. If this bit is set to 1, the character reception completion interrupt will be generated at the end of the character reception.

Bit 0 **PARE**: Parity error interrupt control
0: Disable
1: Enable
This bit is set and cleared by application program and is used to control the parity error interrupt. If this bit is set to 1, the parity error interrupt will be generated when a parity error occurs.

CIPR Register

Bit	7	6	5	4	3	2	1	0
Name	TXBEP	—	IOVFP	VCP	WTP	TXCP	RXCP	PARP
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 **TXBEP**: Transmit buffer empty interrupt pending flag
0: No interrupt pending
1: Interrupt pending
This bit is set by hardware and cleared by a read access to this register using the application program. It is used to indicate whether there is a transmit buffer empty interrupt pending or not. If the corresponding interrupt enable control bit is set to 1 and the transmit buffer is empty, this bit will be set to 1 to indicate that the transmit buffer empty interrupt is pending.

Bit 6 Unimplemented, read as "0"

Bit 5 **IOVFP**: Card current overload interrupt pending flag
0: No interrupt pending
1: Interrupt pending
This bit is set by hardware and cleared by a read access to this register using the application program. It is used to indicate whether there is a card current overload interrupt pending or not. If the corresponding interrupt enable control bit is set to 1 and the card current overloads, this bit will be set to 1 to indicate that the card current overload interrupt is pending.

Bit 4 **VCP**: Card voltage error interrupt pending flag
0: No interrupt pending
1: Interrupt pending
This bit is set by hardware and cleared by a read access to this register using the application program. It is used to indicate whether there is a card voltage error interrupt pending or not. If the corresponding interrupt enable control bit is set to 1 and the card voltage is not in the speficied range, this bit will be set to 1 to indicate that the card voltage error interrupt is pending.

Bit 3 **WTP**: Waiting time counter underflow interrupt pending flag
0: No interrupt pending
1: Interrupt pending
This bit is set by hardware and cleared by a read access to this register using the application program. It is used to indicate whether there is a waiting time counter underflow interrupt pending or not. If the corresponding interrupt enable control bit is set to 1 and the waiting time counter undeflows, this bit will be set to 1 to indicate that the waiting time counter underflow interrupt is pending.

Bit 2 **TXCP**: Character transmission completion interrupt pending flag
0: No interrupt pending
1: Interrupt pending
This bit is set by hardware and cleared by a read access to this register using the application program. It is used to indicate whether there is a character transmission completion interrupt pending or not. If the corresponding interrupt enable control bit is set to 1 and a character has been transmitted, this bit will be set to 1 to indicate that the character transmission completion interrupt is pending.

Bit 1 **RXCP**: Character reception completion interrupt pending flag
0: No interrupt pending
1: Interrupt pending
This bit is set by hardware and cleared by a read access to this register using the application program. It is used to indicate whether there is a character reception completion interrupt pending or not. If the corresponding interrupt enable control bit is set to 1 and a character has been received, this bit will be set to 1 to indicate that the character reception completion interrupt is pending.

Bit 0 **PARP**: Parity error interrupt pending flag
0: No interrupt pending
1: Interrupt pending
This bit is set by hardware and cleared by a read access to this register using the application program. It is used to indicate whether there is a parito error interrupt pending or not. If the corresponding interrupt enable control bit is set to 1 and the parity error occurs, this bit will be set to 1 to indicate that the parito error interrupt is pending.

CTXB Register

Bit	7	6	5	4	3	2	1	0
Name	TB7	TB6	TB5	TB4	TB3	TB2	TB1	TB0
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 **TB7~TB0**: Transmit buffer bit 7 ~ bit 0
The CTXB register is used to store the data to be transmitted.

CRXB Register

Bit	7	6	5	4	3	2	1	0
Name	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
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 **RB7~RB0**: Receive buffer bit 7 ~ bit 0
The CRXB register is used to store the received data. As the character has been received completely, the value in the CRXB register should be read to avoid the next character being overwritten.

Low Voltage Detector – LVD

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

LVD Register

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

LVDC Register

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

Bit 7~6 Unimplemented, read as “0”

Bit 5 **LVDO**: LVD output flag
0: No Low Voltage Detected
1: Low Voltage Detected

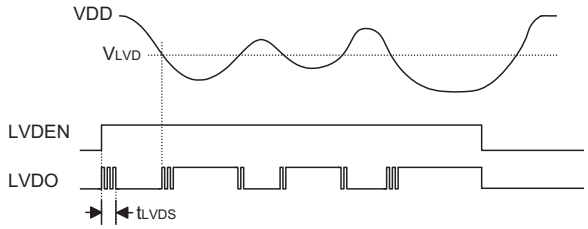
Bit 4 **LVDEN**: Low Voltage Detector Enable control
0: Disable
1: Enable

Bit 3 **VBGEN**: Bandgap Voltage Output Enable control
0: Disable
1: Enable

Bit 2~0 **VLVD2~VLVD0**: LVD Voltage selection
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.0V and 4.0V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is powered down the low voltage detector will remain active if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. When the device is powered down the Low Voltage Detector will remain active if the LVDEN bit is high. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} falls below the preset LVD voltage. This will cause the device to wake-up from the SLEEP or IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before the device enters the SLEEP or IDLE Mode.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. These devices contain several external interrupt and internal interrupts functions. The external interrupts are generated by the action of the external INT0 and INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, LVD, comparators, SPI, I²C, UART, Smart Card interface, USB interface, EEPROM and the A/D converter, etc.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers depends upon the device chosen but fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI3 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
Smart Card operation	CRDE	CRDF	—
USB	USBE	USBF	—
INTn Pins	INTnE	INTnF	n = 0 ~ 1
UART	URnE	URnF	n = 0 ~ 1
Smart Card Insertion/Removal	CRDE	CRDF	—
Multi-function	MFnE	MFnF	n = 0 ~ 3
Time Base	TBnE	TBnF	n = 0 ~ 1
SPI	SPInE	SPInF	n = 0 ~ 1
I²C	IICE	IICF	—
LVD	LVE	LVF	—
Comparator	CPnE	CPnF	n = 0 ~ 1
A/D Converter	ADE	ADF	—
CTM	CTMnPE	CTMnPF	n = 0 ~ 1
	CTMnAE	CTMnAF	
PTM	PTMPE	PTMPF	—
	PTMAE	PTMAF	
STM	STMPE	STMPF	—
	STMAE	STMAF	
EEPROM (HT66F4390)	DEE	DEF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	INT0F	USBF	CRDF	INT0E	USBE	CRDE	EMI
INTC1	SCIRF	UR1F	UR0F	INT1F	CIRE	UR1E	UR0E	INT1E
INTC2	TB1F	TB0F	MF1F	MF0F	TB1E	TB0E	MF1E	MF0E
INTC3	MF3F	MF2F	SPI1F	SPI0F	MF3E	MF2E	SPI1E	SPI0E
MFI0	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
MFI1	STMAF	STMPF	PTMAF	PTMPF	STMAE	STMPE	PTMAE	PTMPE
MFI2 (HT66F4360 /HT66F4370)	—	—	LVF	IICF	—	—	LVE	IICE
MFI2 (HT66F4390)	—	DEF	LVF	IICF	—	DEE	LVE	IICE
MFI3	—	ADF	CP1F	CP0F	—	ADE	CP1E	CP0E

Interrupt Registers List

INTEG Register

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

Bit 7~4 Unimplemented, read as "0"

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

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

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	INT0F	USBF	CRDF	INT0E	USBE	CRDE	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 **INT0F**: INT0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **USBF**: USB interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CRDF**: Smart Card operation interrupt request flag
0: No request
1: Interrupt request

Bit 3 **INT0E**: INT0 interrupt control
0: Disable
1: Enable

Bit 2 **USBE**: USB interrupt control
0: Disable
1: Enable

Bit 1 **CRDE**: Smart Card operation 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	SCIRF	UR1F	UR0F	INT1F	CIRE	UR1E	UR0E	INT1E
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 **SCIRF**: Smart card insertion/removal interrupt request flag
0: No request
1: Interrupt request

Bit 6 **UR1F**: UART1 transfer interrupt request flag
0: No request
1: Interrupt request

Bit 5 **UR0F**: UART0 transfer interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INT1F**: INT1 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **CIRE**: Smart card insertion/removal interrupt control
0: Disable
1: Enable

Bit 2 **UR1E**: UART1 transfer interrupt control
0: Disable
1: Enable

Bit 1 **UR0E**: UART0 transfer interrupt control
0: Disable
1: Enable

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

INTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	TB1F	TB0F	MF1F	MF0F	TB1E	TB0E	MF1E	MF0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **MF1F**: Multi-function 1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **MF0F**: Multi-function 0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

Bit 2 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

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

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

INTC3 Register

Bit	7	6	5	4	3	2	1	0
Name	MF3F	MF2F	SPI1F	SPI0F	MF3E	MF2E	SPI1E	SPI0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **MF3F**: Multi-function 3 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **MF2F**: Multi-function 2 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **SPI1F**: SPI1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **SPI0F**: SPI0 interrupt request flag
0: No request
1: Interrupt request

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

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

Bit 1 **SPI1E**: SPI1 interrupt control
0: Disable
1: Enable

Bit 0 **SPI0E**: SPI0 interrupt control
0: Disable
1: Enable

MFI0 Register

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

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

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

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

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

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

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

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

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

MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	STMAF	STMPF	PTMAF	PTMPF	STMAE	STMPE	PTMAE	PTMPE
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 **STMAF**: STM Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 6 **STMPF**: STM Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **PTMAF**: PTM Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTMPF**: PTM Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3 **STMAE**: STM Comparator A match Interrupt control
0: Disable
1: Enable

Bit 2 **STMPE**: STM Comparator P match Interrupt control
0: Disable
1: Enable

Bit 1 **PTMAE**: PTM Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **PTMPE**: PTM Comparator P match Interrupt control
0: Disable
1: Enable

MFI2 Register – HT66F4360/HT66F4370

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **LVF**: LVD Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **IICF**: I^2C Interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

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

Bit 0 **IICE**: I^2C Interrupt control
0: Disable
1: Enable

MFI2 Register – HT66F4390

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

Bit 7 Unimplemented, read as "0"

Bit 6 **DEF**: EEPROM Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **LVF**: LVD Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **IICF**: I^2C Interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **DEE**: EEPROM Interrupt control
0: Disable
1: Enable

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

Bit 0 **IICE**: I^2C Interrupt control
0: Disable
1: Enable

MFI3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	ADF	CP1F	CP0F	—	ADE	CP1E	CP0E
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 5 **CP1F**: Comparator 1 Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CP0F**: Comparator 0 Interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **ADE**: A/D Converter interrupt control
0: Disable
1: Enable

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

Bit 0 **CP0E**: Comparator 0 Interrupt control
0: Disable
1: Enable

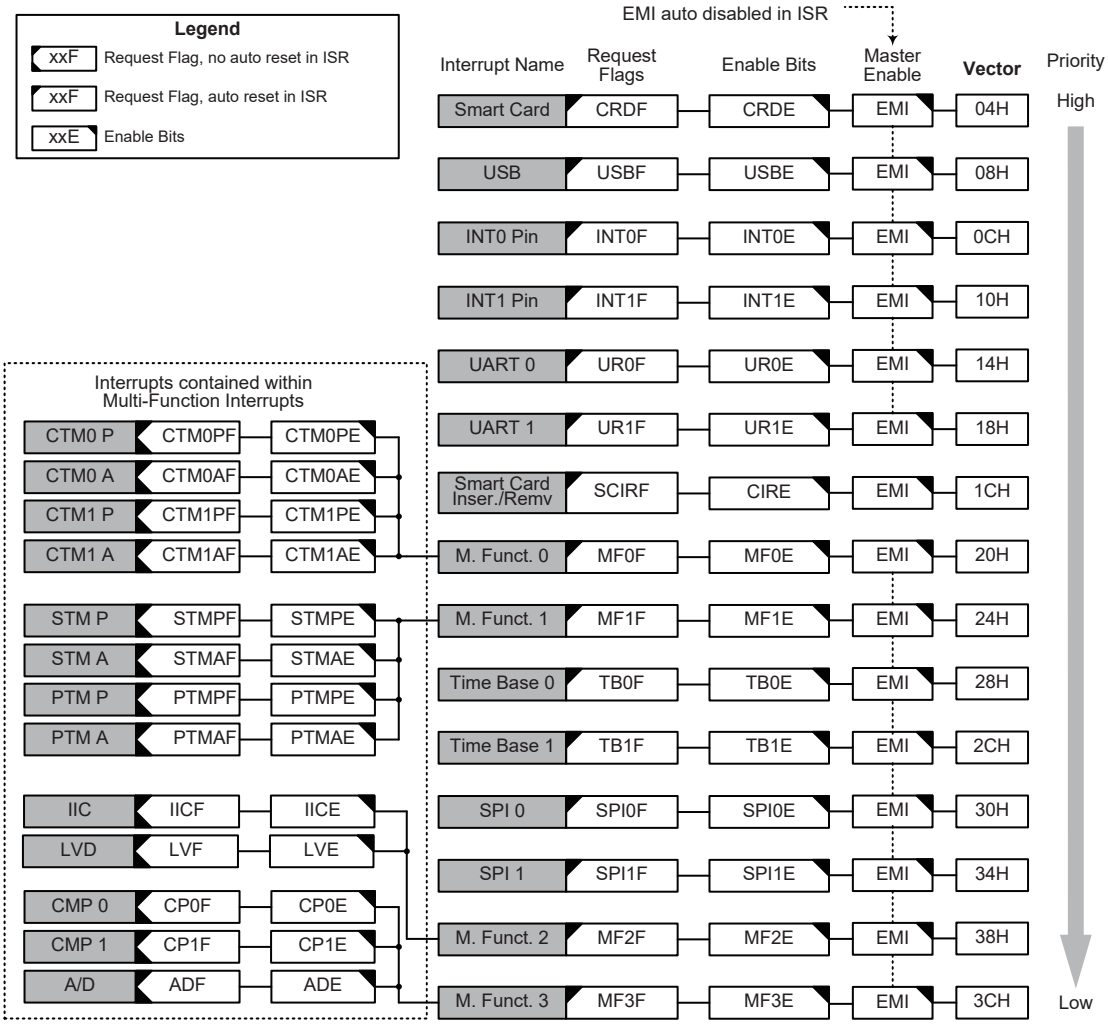
Interrupt Operation

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

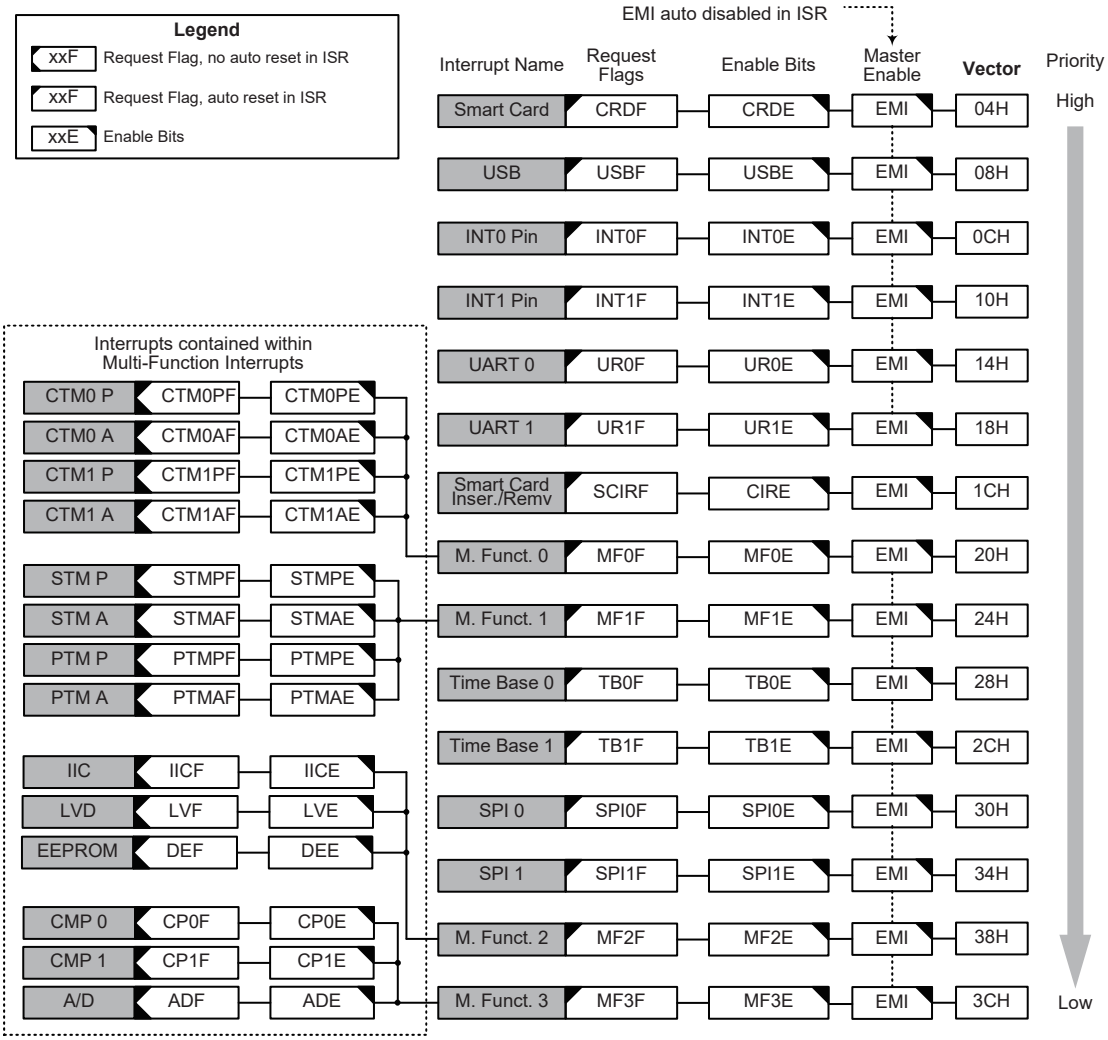
When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will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

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

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



Interrupt Structure – HT66F4360/HT66F4370



Interrupt Structure – HT66F4390

Smart Card Operation Interrupt

For a Smart Card Interrupt to be generated, the global interrupt enable bit EMI must first be set as well as one of the associated Smart Card event Interrupt enable bits. The Smart Card events that will generate an interrupt include situations such as a Card voltage error, a Card current overload, a waiting timer overflow, a parity error, a transmit buffer empty or an end of transmission or reception. Once one of the associated Smart Card event interrupt enable control bits is set, it will automatically set the CRDE bit to 1 to enable the related Smart Card interrupt. An actual Smart Card Interrupt will take place when the Smart Card Interrupt request flag, CRDF, is set, a situation that will occur when a Smart Card event has occurred. When the interrupt is enabled, the stack is not full and a Smart Card event has occurred, a subroutine call to the Smart Card interrupt vector will take place. When the Smart Card Interrupt is serviced, the Smart Card interrupt request flag CRDF will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

USB Interrupt

A USB interrupt request will take place when the USB interrupt request flags, USBF, is set, a situation that will occur when an endpoint is accessed. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and USB interrupt enable bit, USBE, must first be set. When the interrupt is enabled, the stack is not full and an endpoint is accessed, a subroutine call to the USB interrupt vector will take place. When the interrupt is serviced, the USB interrupt request flag, USBF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

External Interrupt

The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

UART Transfer Interrupt

The UARTn Transfer Interrupt is controlled by several UARTn transfer conditions. 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 data available, receiver overrun, address detect and an RXn pin wake-up. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and UARTn Interrupt enable bit, URnE, 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 UARTn Interrupt vector will take place. When the interrupt is serviced, the UARTn Interrupt flag, URnF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

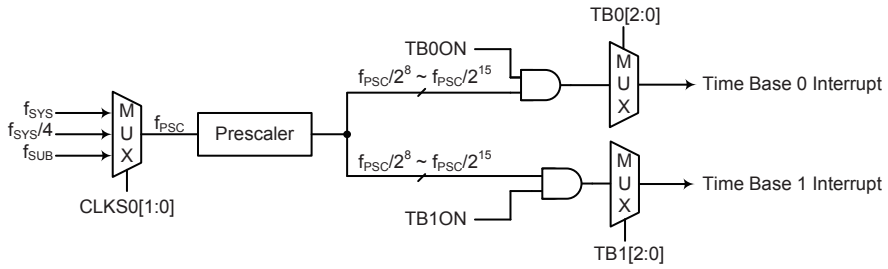
Smart Card Insertion/Removal Interrupt

For a Smart Card Insertion/Removal Interrupt to be generated, the global interrupt enable bit EMI and the Smart Card Insertion/Removal interrupt enable bit CIRE must first be set. An actual Smart Card Insertion/Removal Interrupt will take place when the Smart Card Insertion/Removal Interrupt request flag, SCIRF, is set, a situation that will occur when the Smart Card has been inserted or removed. When the interrupt is enabled, the stack is not full and the Smart Card has been inserted or removed, a subroutine call to the Smart Card Insertion/Removal Interrupt vector will take place. When the Smart Card Insertion/Removal Interrupt is serviced, the Smart Card Insertion/Removal interrupt request flag SCIRF will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Time Base Interrupt

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

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, f_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKS01 and CLKS00 bits in the PSCR register respectively.



Time Base Interrupts

PSCR Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKS01~CLKS00**: Prescaler clock source selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

TB0C Register

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

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

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Time Base 0 time-out period selection
000: $2^8/f_{PSC}$
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 Enable Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB12~TB10**: Time Base 1 time-out period selection
000: $2^8/f_{PSC}$
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: $2^{15}/f_{PSC}$

SPI Interrupt

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

Multi-function Interrupt

Within the device there are up to four Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts, I^2C interface interrupts, LVD interrupt, EEPROM interrupt, comparator interrupt and A/D converter interrupt.

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

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

TM Interrupt

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

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

I²C Interrupt

The I²C Interface Interrupt is contained within the Multi-function Interrupt. An I²C Interrupt request will take place when the I²C Interrupt request flag, IICF, is set, which occurs when an I²C slave address match, a byte of data has been received or transmitted by the I²C interface or an I²C bus time-out occurrence. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, the associated Multi-function interrupt enable bit 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 respective Multi-function Interrupt vector will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the Multi-function interrupt request flag will automatically be cleared. As the I²C Interrupt request flag, IICF, will not be automatically cleared, it has to be cleared by the application program.

LVD Interrupt

The Low Voltage Detector Interrupt is contained within the Multi-function Interrupt. An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the Multi-function Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the Multi-function interrupt request flag will be automatically cleared. As the LVF flag will not be automatically cleared, it has to be cleared by the application program.

EEPROM Interrupt

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

Comparator Interrupt

The Comparator Interrupt is contained within the Multi-function Interrupt. A Comparator Interrupt request will take place when the Comparator Interrupt request flag, CPnF, is set, which occurs when the Comparator n output changes state. To allow the program to branch to its respective Multi-function interrupt vector address, the global interrupt enable bit, EMI, Comparator Interrupt enable bit, CPnE, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and the Comparator n output changes state, a subroutine call to the respective Multi-function Interrupt vector will take place. When the Comparator n Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the Multi-function interrupt request flag will be automatically cleared. As the CPnF flag will not be automatically cleared, it has to be cleared by the application program.

A/D Converter Interrupt

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

Interrupt Wake-up Function

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

Programming Considerations

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

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

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

Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

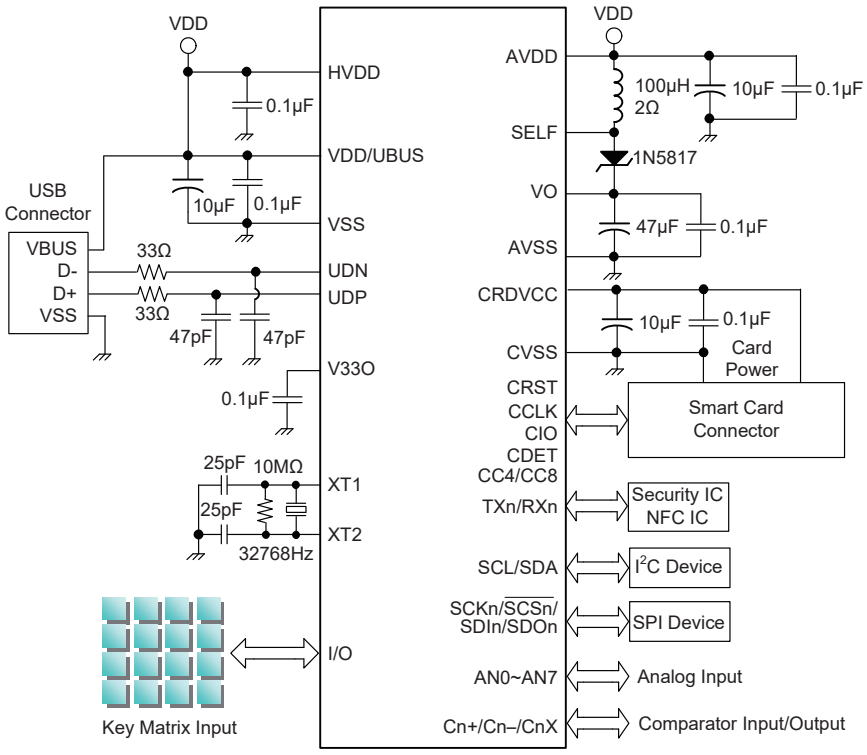
To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later as the application software has no control over the configuration options. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
1	High Speed Crystal oscillator frequency selection 12MHz or 6MHz

Application Circuits



HT66F4360/HT66F4370/HT66F4390

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 three 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 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 set 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 to TBLH and Data Memory	2Note	None
ITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

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

Extended Instruction Set

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

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

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

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

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation: Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s): None

CLR [m] Clear Data Memory

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

Operation: [m] ← 00H

Affected flag(s): None

CLR [m].i Clear bit of Data Memory

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

Operation: [m].i ← 0

Affected flag(s): None

CLR WDT Clear Watchdog Timer

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

Operation: WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CPL [m] Complement Data Memory

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

Operation: [m] ← $\overline{[m]}$

Affected flag(s): Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation: ACC ← $\overline{[m]}$

Affected flag(s): Z

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

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

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

Affected flag(s): C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC "OR" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s): None

RLC [m] Rotate Data Memory left through Carry

Description: The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

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

Affected flag(s): C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s): C

RR [m] Rotate Data Memory right

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

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

Affected flag(s): None

RRA [m] Rotate Data Memory right with result in ACC

Description: Data in the specified Data Memory and the carry flag are 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] - $\overline{C}$

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

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: [m] ← ACC − [m] − C

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

SDZ [m] Skip if decrement Data Memory is 0

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation:
[m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation:
ACC ← [m] − 1
Skip if ACC=0

Affected flag(s): None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

SNZ [m].i Skip if Data Memory is not 0

Description If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

Description If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m]≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

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

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

Description If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

TABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer pair (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRD [m] Increment table pointer low byte first and read table to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Extended Instruction Definition

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

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

LADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

LADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″AND″ [m]

Affected flag(s) Z

LCLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

LCLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

LCPL [m] Complement Data Memory

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

Operation $[m] \leftarrow \overline{[m]}$

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

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

Operation $ACC \leftarrow \overline{[m]}$

Affected flag(s) Z

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

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

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

LDEC [m] Decrement Data Memory

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

Operation $[m] \leftarrow [m] - 1$

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

LINC [m] Increment Data Memory

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

Operation $[m] \leftarrow [m] + 1$

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] + 1$

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

LMOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

LRL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

LRLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

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

Affected flag(s) C

LRLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) C

LRR [m] Rotate Data Memory right

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

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

Affected flag(s) None

LRRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory and the carry flag are 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 two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

LSET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

LSET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

LSIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

LSIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

LSNZ [m].i Skip if Data Memory is not 0

Description If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

Description If the content of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m] ≠ 0

Affected flag(s) None

LSUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

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

LSUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

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

LSWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

LSWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

LSZ [m] Skip if Data Memory is 0

Description If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

LSZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

LSZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

LTABRD [m] Read table (current page) to TBLH and Data Memory

Description The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LTABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRD [m] Increment table pointer low byte first and read table to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LXOR A,[m] Logical XOR Data Memory to ACC

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

LXORM A,[m] Logical XOR ACC to Data Memory

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

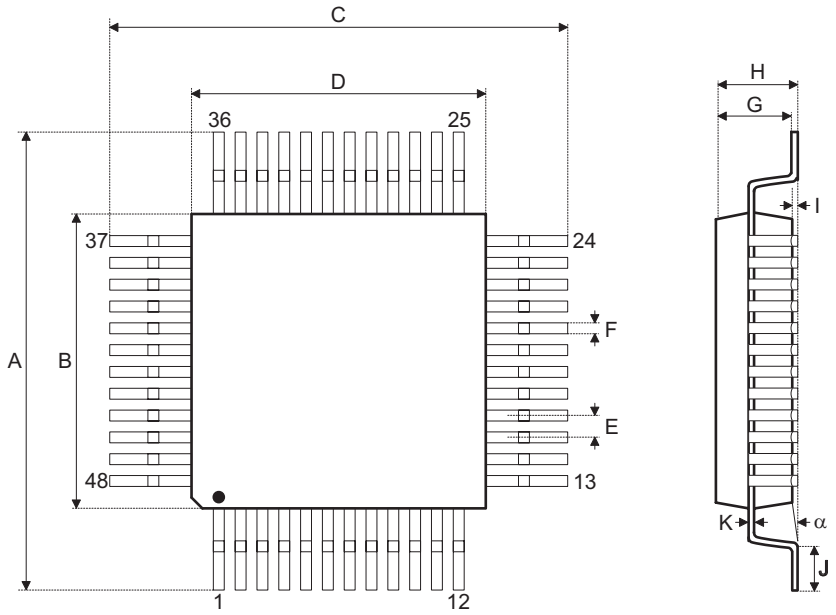
Package Information

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

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

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

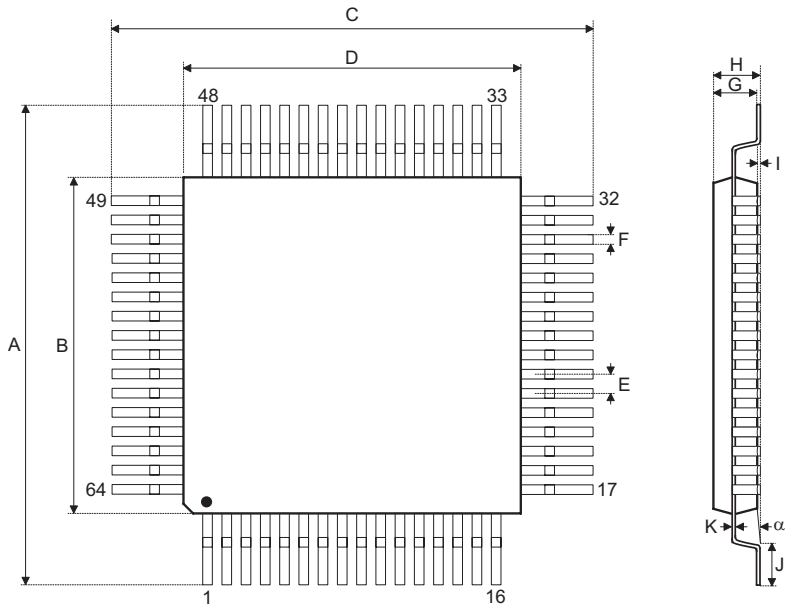
48-pin LQFP (7mm×7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.020 BSC	—
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.50 BSC	—
F	0.17	0.22	0.27
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

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



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.016 BSC	—
F	0.005	0.007	0.009
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.40 BSC	—
F	0.13	0.18	0.23
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°