USB Low Speed Flash Type MCU

HT68FB240

Revision: V1.00 Date: April 18, 2014

www.holtek.com

Table of Contents

Features

CPU Features

- Operating voltage:
  - V_{DD} (MCU)
    f_{SYS} = 6MHz: 2.2V~5.5V
    f_{SYS} = 12MHz: 3.3V~5.5V
  - V_{DD} (USB mode)
    f_{SYS} = 6MHz/12MHz: 3.3V~5.5V
    f_{SYS} = 16MHz: 4.5V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- HALT function and wake-up feature reduce power consumption
- Oscillators:
  - Internal 12MHz RC – HIRC
  - Internal 32kHz RC – LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 4K × 16
- RAM Data Memory: 160 × 8
- USB 2.0 Low Speed compatible
- Up to 3 endpoints supported including endpoint 0
- All endpoints except endpoint 0 support interrupt transfer
- Endpoint 0 supports control transfer for Setup, In and Out token respectively
- Endpoint 0 has 8 bytes FIFO size, supports DMA interface for configure USB descriptor
- Support 3.3V LDO and integrate an internal 1.5K ohm pull-up resistor on UDN line
- Internal 12MHz RC OSC with 1.5% accuracy for all USB modes
- Watchdog Timer function
- Up to 34 bidirectional I/O lines
- Two pin-shared external interrupts
- Two Compact Timer Modules for time measure, compare match output, PWM output functions
- Serial Interface Modules with SPI and I^2C interfaces
- Low voltage reset function
- Low voltage detect function
- In-system programmable in 32 words sector
- Support 3 channels 8-bit PWM
- Support ISP/IAP function
- Package types: 46-pin QFN, 48-pin LQFP
- OCDS debug Interface for HT68VB240 only

General Description

The HT68FB240 is Flash Memory I/O with USB type 8-bit high performance RISC architecture microcontrollers, designed for applications that interface directly to analog signals and which require an USB interface. Offering users the convenience of Flash Memory multi-programming features, these devices also include a wide range of functions and features. Other memory includes an area of RAM Data Memory.

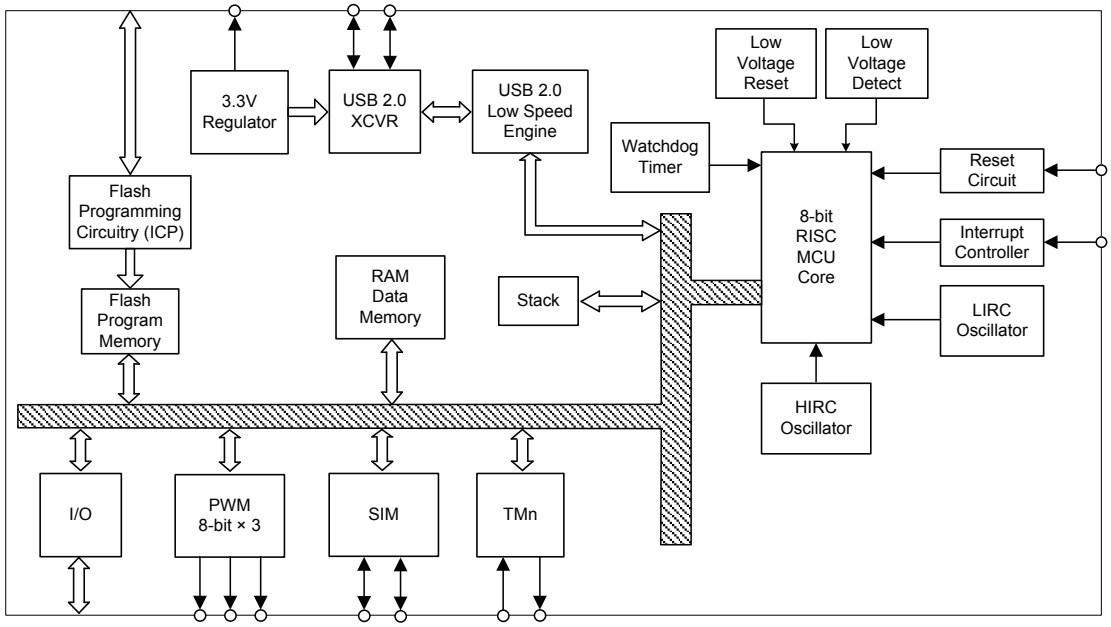
Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI, I^2C and USB interface functions, three popular interfaces which provide designers with a means of easy communication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

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

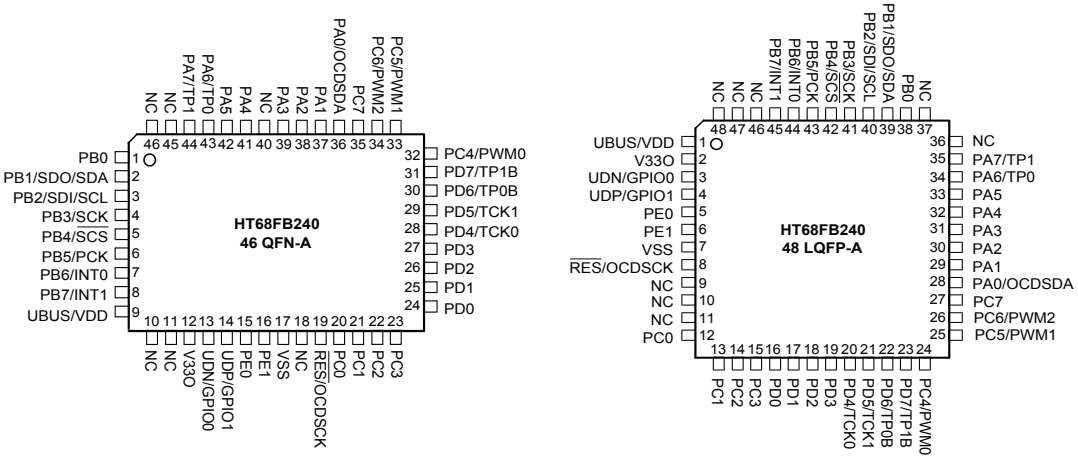
The inclusion of flexible I/O programming features along with many other features ensure that the devices will find specific excellent use in a wide range of application possibilities such as sensor signal processing, motor driving, industrial control, consumer products, subsystem controllers, etc.

This device is fully supported by the Holtek range of fully functional development and programming tools, providing a means for fast and efficient product development cycles.

Block Diagram



Pin Assignment



Pin Description

The pins on this device 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 Serial Port pins etc. The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/ OCDSDA	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	OCDSDA	—	ST	CMOS	OCDS data input/output, for EV chip only.
PA1~PA5	PAn	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
PA6/TP0	PA6	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	TP0	TMPC	ST	CMOS	TM0 output
PA7/TP1	PA7	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	TP1	TMPC	ST	CMOS	TM1 output
PB0	PB0	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
PB1/SDO/ SDA	PB1	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	SDO	—	—	CMOS	SPI serial data output
	SDA	—	ST	NMOS	I²C data line
PB2/SDI/SCL	PB2	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	SDI	—	ST	—	SPI serial data input
	SCL	—	ST	NMOS	I²C clock line
PB3/SCK	PB3	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	SCK	—	ST	CMOS	SPI serial clock

Pin Name	Function	OPT	I/T	O/T	Description
PB4/$\overline{SCS}$	PB4	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	$\overline{SCS}$	—	ST	CMOS	SPI slave select pin
PB5/PCK	PB5	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	PCK	—	—	CMOS	Peripheral output clock
PB6/INT0	PB6	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	INT0	—	ST	—	External interrupt 0
PB7/INT1	PB7	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	INT1	—	ST	—	External interrupt 1
PC0~PC3	PCn	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
PC4/PWM0	PC4	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	PWM0	PWMC	—	CMOS	PWM0 output
PC5/PWM1	PC5	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	PWM1	PWMC	—	CMOS	PWM1 output
PC6/PWM2	PC6	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	PWM2	PWMC	—	CMOS	PWM2 output
PC7	PC7	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
PD0~PD3	PDn	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
PD4/TCK0	PD4	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	TCK0	—	ST	—	TM0 clock input
PD5/TCK1	PD5	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	TCK1	—	ST	—	TM1 clock input
PD6/TP0B	PD6	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	TP0B	TMPC	ST	CMOS	TM0 inverter output
PD7/TP1B	PD7	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	TP1B	TMPC	ST	CMOS	TM1 inverter output
PE0~PE1	PEn	PXPU PXWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
$\overline{RES}$/ OCDSCK	$\overline{RES}$	—	ST	—	Reset input
	OCDSCK	—	ST	—	OCDS clock input, for EV chip only.
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
UBUS/VDD	UBUS	—	PWR	—	USB SIE VDD
	VDD	—	PWR	—	Power supply

Pin Name	Function	OPT	I/T	O/T	Description
V33O	V33O	—	—	PWR	3.3V regulator output
VSS	VSS	—	PWR	—	Ground

Note: I/T: Input type; O/T: Output type
OP: Optional by configuration option (CO) or register option
PWR: Power; CO: Configuration option
ST: Schmitt Trigger input; CMOS: CMOS output

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}−0.3V to V_{SS}+6.0V
Input Voltage ..... V_{SS}−0.3V to V_{DD}+0.3V
Storage Temperature ..... -50˚C to 125˚C
Operating Temperature ..... -40˚C to 85˚C
I_{OH} Total ..... -100mA
I_{OL} Total 150mA
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 (HIRC OSC)	—	f_{SYS}=12MHz	3.3	—	5.5	V
I_{DD1}	Operating Current (HIRC OSC, $f_{SYS}=f_H$, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, f_H=12MHz, WDT enable USB disable, LVR enable	—	2.2	3.3	mA
		5V		—	5.0	7.5	mA
I_{DD2}	Operating Current (LIRC OSC, $f_{SYS}=f_L=f_{LIRC}$, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, WDT enable, f_{LIRC}=32K, Clear CLK_ADJ (SYSC.7=0) USB disable, LVR enable	—	35	70	μA
		5V		—	70	100	μA
I_{DD3}	Operating Current (HIRC OSC, $f_{SYS}=f_H$, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, f_H=12MHz, WDT enable, USB enable, V33O on, LVR enable	—	4.8	10	mA
		5V		—	11	16	mA
I_{STB1}	Standby Current (Idle 0) (HIRC OSC, f_{SYS}=off, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, system HALT, WDT enable, f_{SYS}=12MHz off (FSYSON=0), Clear CLK_ADJ (SYSC.7=0)	—	1.5	3.0	μA
		5V		—	3.0	6.0	μA
I_{STB2}	Standby Current (Sleep 0) (HIRC OSC, f_{SYS}=off, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, system HALT, WDT disable, f_{SYS}=12MHz, Clear CLK_ADJ (SYSC.7=0)	—	0.1	1	μA
		5V		—	0.3	2	μA
I_{SUS1}	Suspend Current (Sleep 0) (HIRC OSC, f_{SYS}=off, $f_S=f_{SUB}=f_{LIRC}$)	5V	No load, system HALT, WDT disable, LVR disable USB transceiver, 3.3V Regulator on, clr SUSP2 (UCC.4) and clr RCTRL (UCC.7),Clear CLK_ADJ (SYSC.7=0)	—	330	400	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{SUS2}	Suspend Current (Sleep 0) (HIRC OSC, f_{SYS}=off, $f_S=f_{SUB}=f_{LIRC}$)	5V	No load, system HALT, WDT disable, LVR disable USB transceiver, 3.3V Regulator off, set SUSP2 (UCC.4) and set RCTRL (UCC.7),Clear CLK_ADJ (SYSC.7=0)	—	220	300	μA
V_{IL1}	Input Low Voltage for I/O Ports, TCK and INT	5V	—	0	—	$0.3V_{DD}$	V
		—	(Except V_{DD}=5V)	0	—	$0.2V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TCK and INT	5V	—	$0.7V_{DD}$	—	V_{DD}	V
		—	(Except V_{DD}=5V)	$0.8V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	I/O Port Sink Current	5V	V_{OL}=0.4V	2	4	—	mA
I_{OH}	I/O Port Source Current	5V	V_{OH}=3.4V	-2	-4	—	mA
V_{V33O}	3.3V regulator output	5V	I_{V33O}=20mA	3.0	3.3	3.6	V
R_{UDN}	Pull-high Resistance between UDN and V33O	3.3V	—	-5%	1.5	+5%	kΩ
R_{PH}	Pull-high Resistance of I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

A.C. Characteristics

Ta= 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System clock (HIRC OSC) USB mode (USB On)	4.2V~5.5V	Ta=25°C	-3%	12	+3%	MHz
f_{LIRC}	System clock (32K RC)	5V	Ta=25°C	-10%	32	+10%	kHz
		2.2V~5.5V	Ta=-40°C to 85°C	-50%	32	+60%	kHz
t_{TCK}	TCKn Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{RES}	External Reset Minimum Low Pulse Width	—	—	10	—	—	μs
t_{INT}	Interrupt Minimum Pulse Width	—	—	10	—	—	μs
t_{SST}	System Start-up Timer Period (Wake-up from HALT, f_{SYS} off at HALT state, Slow Mode → Normal Mode)	—	f_{SYS}=HIRC	1024	—	—	t_{SYS}
		—	f_{SYS}=LIRC	2	—	—	t_{SYS}
	System Start-up Timer Period (Wake-up from HALT, f_{SYS} on at HALT state)	—	—	2	—	—	t_{SYS}
t_{RSTD}	System Reset Delay Time (Power On Reset, LVR reset, LVR S/W (LVRC) reset, WDT S/W (WDTC) reset)	—	—	25	50	100	ms
	System Reset Delay Time (Any Reset except Power On Reset, LVR reset, LVR S/W (LVRC) reset, WDT S/W (WDTC) reset)	—	—	8.3	16.7	33.3	ms

LVD & LVR Electrical Characteristics

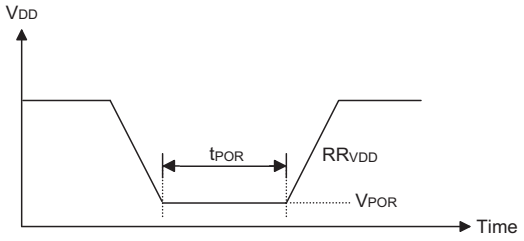
Ta= 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR1}	Low Voltage Reset Voltage	—	LVR Enable	-5%	2.1	+5%	V
V_{LVD1}	Low Voltage Detector Voltage	—	LVDEN = 1, V_{LVD} = 2.0V	-5%	2.0	+5%	V
V_{LVD2}			LVDEN = 1, V_{LVD} = 2.2V		2.2		V
V_{LVD3}			LVDEN = 1, V_{LVD} = 2.4V		2.4		V
V_{LVD4}			LVDEN = 1, V_{LVD} = 2.7V		2.7		V
V_{LVD5}			LVDEN = 1, V_{LVD} = 3.0V		3.0		V
V_{LVD6}			LVDEN = 1, V_{LVD} = 3.3V		3.3		V
V_{LVD7}			LVDEN = 1, V_{LVD} = 3.6V		3.6		V
V_{LVD8}			LVDEN = 1, V_{LVD} = 4.0V		4.0		V
I_{LVR}	Additional Power Consumption if LVR is used	3V	LVR disable → LVR enable	—	30	45	μA
		5V		—	60	90	μA
I_{LVD}	Additional Power Consumption if LVD is used	3V	LVD disable → LVD enable (LVR disable)	—	40	60	μA
		5V		—	75	115	μA
		3V	LVD disable → LVD enable (LVR enable)	—	30	45	μA
		5V		—	60	90	μA
t_{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Low Voltage Width to Interrupt	—	—	20	45	90	μs
t_{SRESET}	Software Reset Width to Reset	—	—	45	90	120	μs
t_{LVDS}	LVDO stable time	—	For LVR enable, LVD off → on	—	—	15	μs
		—	For LVR disable, LVD off → on	—	—	15	μs

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_{VDD}	V_{DD} Rise 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

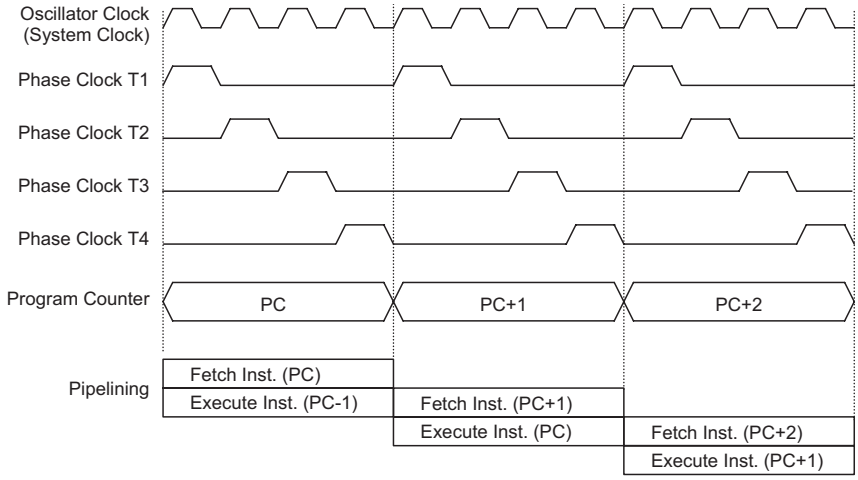


System Architecture

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

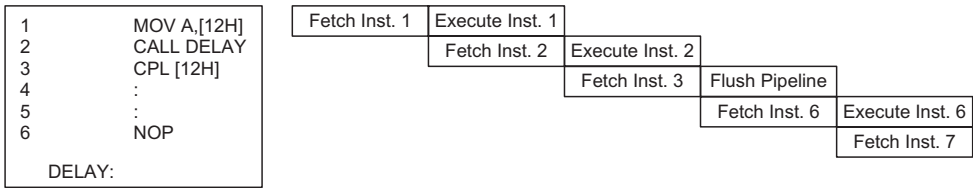
Clocking and Pipelining

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



System Clocking and Pipelining

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



Instruction Fetching

Program Counter

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

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

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

Program Counter

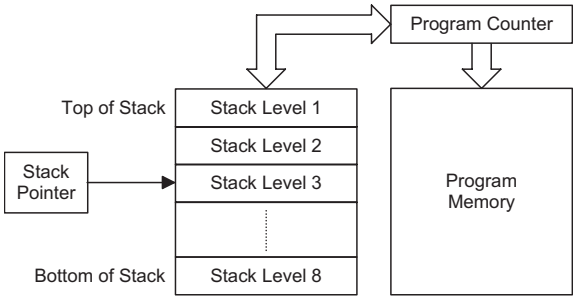
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly. However, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 8 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



Arithmetic and Logic Unit – ALU

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

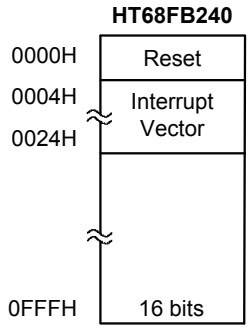
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement INCA, INC, DECA, DEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

Flash Program Memory

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

Structure

The Program Memory has a capacity of 4K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



Program Memory Structure

Special Vectors

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

Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the "TABRD [m]" or "TABRDL [m]" instructions, respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as 0.

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

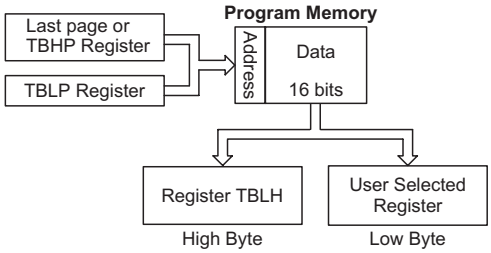


Table Program Example

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

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

Table Read Program Example

```
tempreg1 db ?          ; temporary register #1
tempreg2 db ?          ; temporary register #2
:
mov a, 06h             ; initialise low table pointer - note that this address
mov tblp, a            ; is referenced
mov a, 0Fh             ; initialise high table pointer
mov tbhp, a
:
tabrd tempreg1         ; transfers value in table referenced by table pointer data at
program
                       ; memory address F06H transferred to tempreg1 and TBLH
dec tblp               ; reduce value of table pointer by one
tabrd tempreg2         ; transfers value in table referenced by table pointer data at
program
                       ; memory address F05H transferred to tempreg2 and TBLH in this
                       ; example the data 1AH is transferred to tempreg1 and data 0FH to
                       ; register tempreg2
:
org F00h               ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

In System Programming – ISP

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-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 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 of the devices 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.

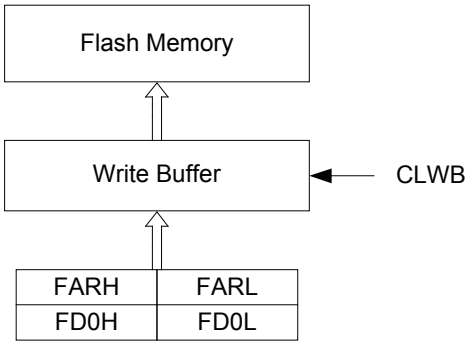
Flash Memory Read/Write Page Size

The Flash memory page size is 32 words. The page and buffer size are assigned as 32 words.

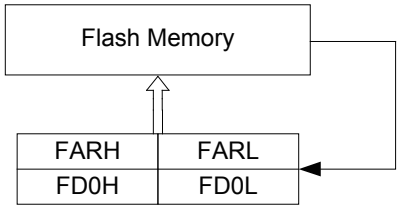
The following diagram illustrates the Read/Write page and buffer assignment. The write buffer is controlled by the CLWB bit in the FRCR register. The CLWB bit can be set high to enable the Clear Write Buffer procedure, as the procedure is finished, this bit will be cleared to low by hardware.

The Write Buffer is filled when the FWEN bit is set to high, when this bit is set high, the data in the Write buffer will be written to the Flash ROM, the FWT bit is used to indicate the writing procedure. Setting this bit high and check if the write procedure is finished, this bit will be cleared by hardware. The Read Byte can be assigned by the address. The FDEN is used to enable the read function and the FRD is used to indicate the reading procedure. When the reading procedure is finished, this bit will be cleared by hardware.

Device	Page Size (Words)	Write Buffer (Words)
HT68FB240 (4K×16)	32	32



Write one word to FD0L/FD0H



Read one word to FD0L/FD0H

Note: 1. Writing a data into high byte, which means the High/Low byte Data is written into Write Buffer, will cause the Flash memory address increased by one automatically and the new address will be loaded to the FARH and FARL registers. However, the user can also fill the new address by filling the data into FARH and FARL registers in the same page, then the data will be written into the corresponding address.

2. If the address already reached the boundary of the flash memory, such as 11111b of the 32 words, at this moment, the address will not be increased and the address will stop at the last address of that page and the writing data is invalid.

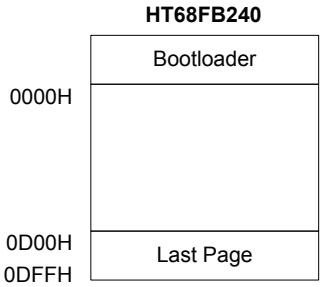
3. At this point, the user has to set a new address again to fill a new data.

4. If the data is writing using the write buffer, the write buffer will be cleared by hardware automatically after the write procedure is ready in 2ms.

5. First time use the Write buffer or renew the data in the Write buffer, the user can use to Clear buffer bit (CLWB) to clear write buffer.

ISP Bootloader

The devices provide the ISP Bootloader function to upgrade the software in the Flash memory. The user can select to use the ISP Bootloader application software provided by Holtek IDE tool or to create his own Bootloader software. When the Holtek Bootloader software is selected, that will occupy 0.5K words area in the Flash memory. The accompanyimg diagram illustrates the Flash memory structure with Holtek Bootloader software.



Flash Program Memory Registers

There are two address registers, four 16-bit data registers and one control register. The control register is located in Bank 1 and the other registers are located in Bank 0. Read and Write operations to the Flash memory are carried out in 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 single control register is named FCR. As the FARL and FDnL registers are located in Bank 0, they can be directly accessed in the same was as any other Special Function Register. The FARH, FDnH, FCR and FRCR registers however, being located in Bank 1, cannot be addressed directly and can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1.

Program Memory Register List

Register Name	Bit							
	7	6	5	4	3	2	1	0
FARL	D7	D6	D5	D4	D3	D2	D1	D0
FARH	—	—	—	—	D11	D10	D9	D8
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
FCR	CFWEN	FMOD2	FMOD1	FMOD0	BWT	FWT	FRDEN	FRD
FRCR	—	—	—	FSWRST	—	—	—	CLWB

• **FARL Register**

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

"x" unknown

Bit 7~0 **D7~D0:** Flash Program Memory address
Flash Program Memory address bit 7 ~ bit 0

• **FARH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	x	x	x	x

"x" unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **D11~D8:** Flash Program Memory address
Flash Program Memory address bit 11 ~ 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	x	x	x	x	x	x	x	x

"x" unknown

Bit 7~0 **D7~D0:** The first Flash ROM data
The first Flash ROM data 7~0

• **FD0H Register**

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

"x" unknown

Bit 7~0 **D15~D8:** The first Flash ROM data
The first Flash ROM data 15~8

• **FD1L Register**

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

"x" unknown

Bit 7~0 **D7~D0:** The second Flash ROM data
The second Flash ROM data 7~0

• **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	x	x	x	x	x	x	x	x

"x" unknown

Bit 7~0 **D15~D8:** The second Flash ROM data
The second Flash ROM data 15~8

• **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	x	x	x	x	x	x	x	x

"x" unknown

Bit 7~0 **D7~D0:** The third Flash ROM data
The third Flash ROM data 7~0

• **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	x	x	x	x	x	x	x	x

"x" unknown

Bit 7~0 **D15~D8:** The third Flash ROM data
The third Flash ROM data bit 15~8

• **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	x	x	x	x	x	x	x	x

"x" unknown

Bit 7~0 **D7~D0:** The fourth Flash ROM data
The fourth Flash ROM data 7~0

• **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	x	x	x	x	x	x	x	x

"x" unknown

Bit 7~0 **D15~D8:** The fourth Flash ROM data
The fourth Flash ROM data 15~8

• **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 ROM Write Enable bit, FWEN, control bit
0: disable
1: unimplemented
This bit is used to control the FWEN bit enable or disable. When this bit is cleared to low by software, the Flash memory write enable control bit, FWEN will be cleared to low as well. It's ineffective to set this bit to high. The user can check this bit to confirm the FWEN status.

Bit 6~4 **FMOD2~FMOD0:** Flash Program memory, Configuration option memory operating mode control bits
000: write memory mode
001: page erase mode
010: reserved
011: read memory mode
100: reserved
101: reserved
110: FWEN (flash memory write enable) bit control mode
111: reserved

Bit 3 **BWT:** Mode change control
0: mode change cycle has finished
1: activate a mode change cycle
This bit will be automatically reset to zero by the hardware after the mode change cycle has finished.

Bit 2 **FWT:** Flash memory Write Control
0: write cycle has finished
1: activate a write cycle
This is the Flash memory 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.

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

Bit 0 **FRD:** Flash memory Read Control
0: read cycle has finished
1: activate a read cycle
This is the Flash memory 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 FRDEN has not first been set high.
Note: The FWT, FRDEN and FRD bits can not be set to "1" at the same time with a single instruction.

- **FRCR Register**

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

Bit 7~5 Unimplemented, read as "0"

Bit 4 **FSWRST:** Software Reset MCU control bit
Described elsewhere

Bit 3~1 unimplemented, read as "0"

Bit 0 **CLWB:** Flash Program memory Write buffer clear control bit
0: do not initiate clear Write Buffer or clear process
1: initiate clear Write Buffer process
This bit is used to control the Flash Program memory clear Write buffer process. It will be set by software and cleared by hardware.

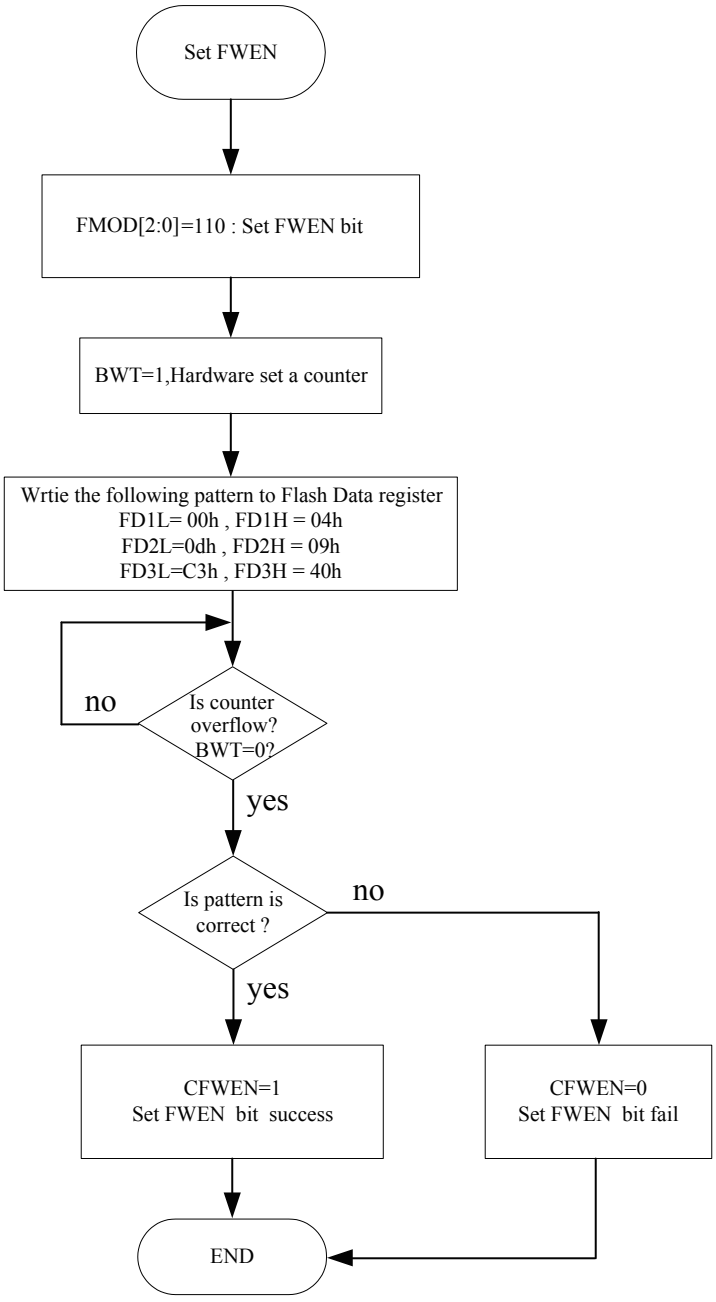
In Application Program – IAP

Offering users the convenience of Flash Memory multi-programming features, the device not only provides an ISP function, but also an additional IAP function. 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 SPI, using I/O pins. Designers can assign I/O pins to communicate with the external memory device, including the updated program. 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 IAP firmware.

Enable Flash Write Control Procedure

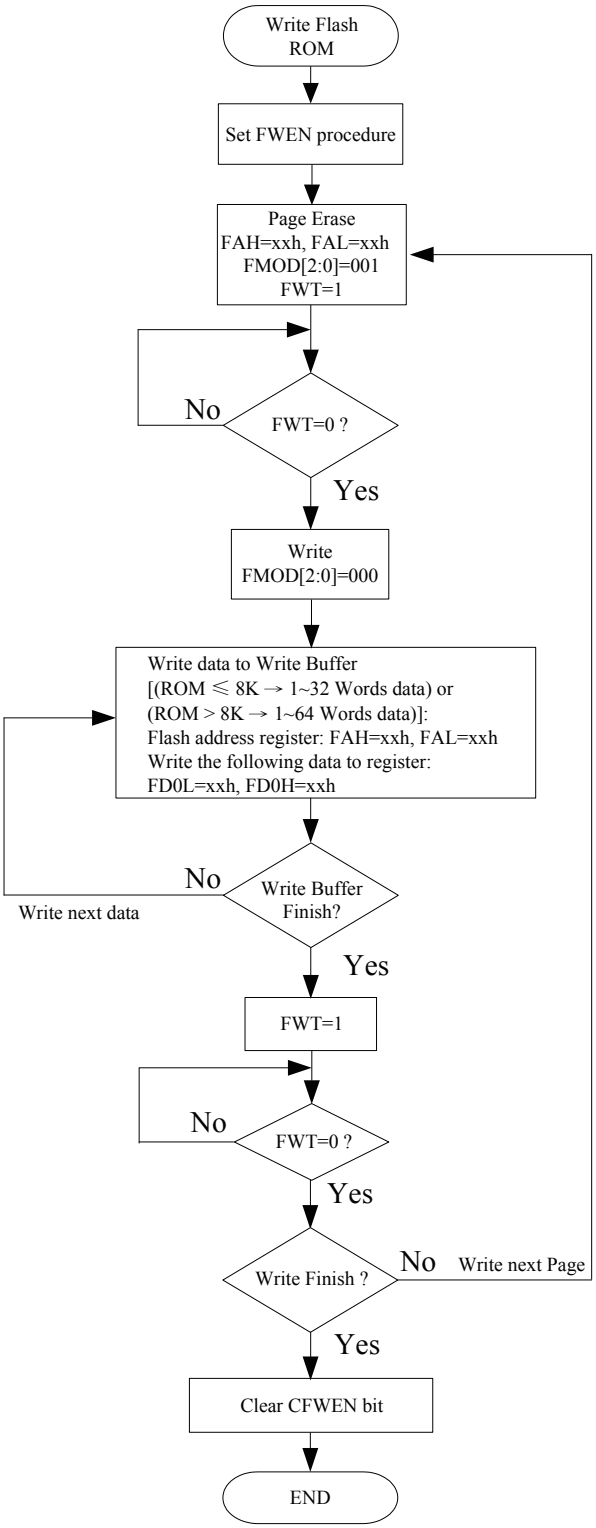
The first procedure to implement the IAP firmware is to enable the Flash Write control which includes the following steps.

- Write data "110" to the FMOD[2:0] bits in the FCR register to enable the Flash write control bit, FWEN.
- Set the BWT bit in the FCR register to "1".
- The device will start a 1ms counter. The user should write the correct data pattern into the Flash data registers, namely FD1L~FD3L and FD1H~FD3H, during this period of time.
- Once the 1ms counter has overflowed or if the written pattern is incorrect, the enable Flash write control procedure will be invalid and the user should repeat the above procedure.
- No matter whether the procedure is valid or not, the devices will clear the BWT bit automatically.
- The enable Flash write pattern data is (00H, 04H, 0DH, 09H, C3H and 40H) and it should be written into the Flash data registers.
- Once the Flash write operation is enabled, the user can update the Flash memory using the Flash control registers.
- To disable the Flash write procedure, the user can only clear the CFWEN bit in the FCR register. There is no need to execute the above procedure.

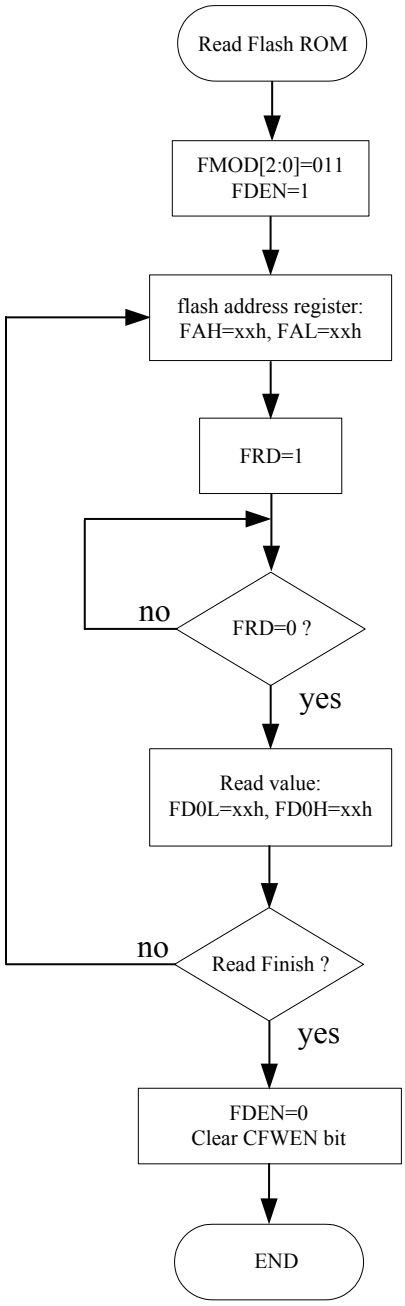

Set FWEN
FMOD[2:0]=110 : Set FWEN bit
BWT=1,Hardware set a counter
Wrtie the following pattern to Flash Data register
FD1L= 00h , FD1H = 04h
FD2L=0dh , FD2H = 09h
FD3L=C3h , FD3H = 40h
no
Is counter overflow? BWT=0?
yes
Is pattern is correct ?
no
yes
CFWEN=1
Set FWEN bit success
CFWEN=0
Set FWEN bit fail
END

Flash Memory Write and Read Procedures

The following flow charts illustrate the Write and Read Flash memory procedures.



Write Flash Program ROM Procedure



Read Flash Program Procedure

In Circuit Programming – ICP

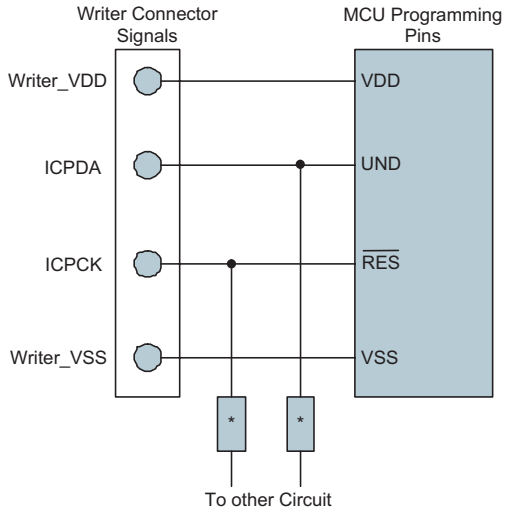
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides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

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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	UDN	Programming Serial Data/Address
ICPCK	$\overline{RES}$	Programming Clock
VDD	VDD	Power Supply
VSS	VSS	Ground

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

During the programming process, taking control of the UDN and $\overline{RES}$ pins for data and clock programming purposes. The user must there take care to ensure that no other outputs are connected to these two pins.



Note: * may be resistor or capacitor. The resistance of * must be greater than 300Ω or the capacitance of * must be less than 1nF.

On-Chip Debug Support – OCDS

An EV chip exists for the purposes of device emulation. This EV chip device also provides an "On-Chip Debug" function to debug the device during the development process. The EV chip and the actual MCU device are almost functionally compatible except for the "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the 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 for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the actual MCU device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For a more detailed OCDS description, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

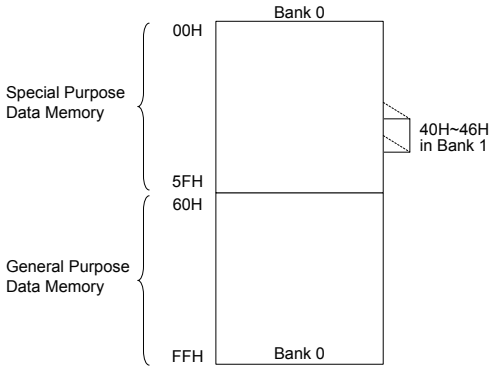
RAM Data Memory

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

Structure

Divided into two sections, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which 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 known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The overall Data Memory is subdivided into two banks. The Special Purpose Data Memory registers are accessible in all banks, with the exception of the FRCR, FCR, FARH and FDnH registers at address from 40H to 46H, which is only accessible in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory for the device is the address 00H.



Data Memory Structure

	Bank 0~1
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	SMOD
0CH	LVDC
0DH	INTEG
0EH	WDTC
0FH	Unused
10H	INTC0
11H	INTC1
12H	INTC2
13H	Unused
14H	MFI0
15H	PAWU
16H	PAPU
17H	PA
18H	PAC
19H	PXWU
1AH	PXPU
1BH	PB
1CH	PBC
1DH	PC
1EH	PCC
1FH	PD
20H	PDC
21H	PE
22H	PEC
23H	Unused
≈	≈
2CH	
2DH	I2CTOC
2EH	SIMC0
2FH	SIMC1
30H	SIMD
31H	SIMA/SIMC2
32H	SBSC

	Bank 0~1	Bank 1
33H	TMPC	
34H	TM0C0	
35H	TM0C1	
36H	TM0DL	
37H	TM0DH	
38H	TM0AL	
39H	TM0AH	
3AH	TM1C0	
3BH	TM1C1	
3CH	TM1DL	
3DH	TM1DH	
3EH	TM1AL	
3FH	TM1AH	
40H	Unused	FRCR
41H	Unused	FCR
42H	FARL	FARH
43H	FD0L	FD0H
44H	FD1L	FD1H
45H	FD2L	FD2H
46H	FD3L	FD3H
47H	USB_STAT	
48H	UINT	
49H	USC	
4AH	UCC	
4BH	AWR	
4CH	STL	
4DH	SIES	
4EH	MISC	
4FH	UFEN	
50H	FIFO0	
51H	FIFO1	
52H	FIFO2	
53H	URDCT	
54H	Unused	
55H		
56H		
57H		
58H	PWMC	
59H	PWM0	
5AH	PWM1	
5BH	PWM2	
5CH	Unused	
5DH	CTRL	
5EH	LVRC	
5FH	SYSC	

: Unused, read as 00H

Special Purpose Data Memory

Special Function Register Description

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

Indirect Addressing Register – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, 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 and IAR1 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 or MP1. Acting as a pair, IAR0 and MP0 can together access data from Bank 0 while the IAR1 and MP1 register pair can access data from any bank. 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, MP1

Two Memory Pointers, known as MP0 and MP1 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 Bank 0, while MP1 and IAR1 are used to access data from all banks according to BP register. Direct Addressing can only be used with Bank 0, all other Banks must be addressed indirectly using MP1 and IAR1.

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

Indirect Addressing Program Example

```
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:
```

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

Bank Pointer – BP

For this device, the Data Memory is divided into two banks, Bank 0 and Bank 1. Selecting the required Data Memory area is achieved using the Bank Pointer. Bit 0 of the Bank Pointer is used to select Data Memory Banks 0~1.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the Power Down Mode, in which case, the Data Memory bank remains unaffected. It should be noted that the Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within any bank. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from Bank 1 must be implemented using Indirect Addressing.

BP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	DMBP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **DMBP0:** Select Data Memory Banks
0: Bank 0
1: Bank 1

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP and TBHP are the table 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), 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 and C 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.

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	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R	R	R/W	R/W	R/W	R/W
POR	—	—	0	0	x	x	x	x

"x" unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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

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

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

Bit 0 **C:** Carry flag
0: no carry-out
1: an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation
C is also affected by a rotate through carry instruction.

Oscillator

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

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer Function. Fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

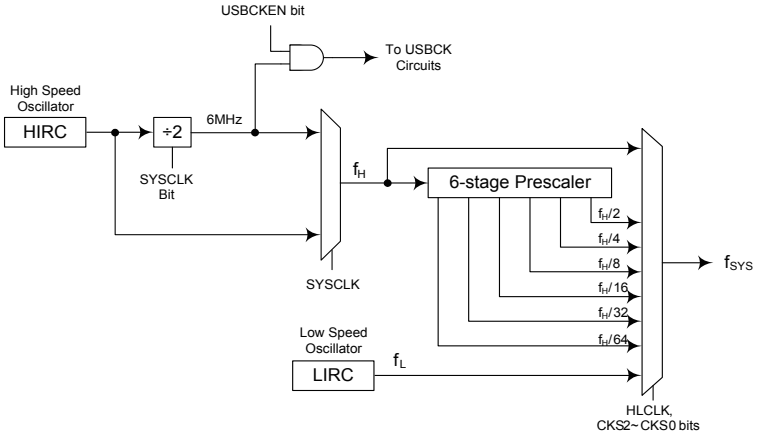
Type	Name	Freq.
Internal High Speed RC	HIRC	12MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are several oscillator sources, a high speed oscillator and a low speed oscillator. The high speed system clock is sourced from the internal 12MHz RC oscillator and the automatic adjust clock. The low speed oscillator is the internal 32kHz RC oscillator. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and CKS2~CKS0 bits in the SMOD register and as the system clock can be dynamically selected.

The actual source clock used for the high speed and the low speed oscillators is chosen via registers. The frequency of the slow speed or high speed system clock is also determined using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator. In addition, the internal automatic adjust frequency clock, whose clock source is supplied by the internal 12MHz RC oscillator, can be enabled by a software control bit to generate various frequencies for the USB interface and system clock.



System Clock Configurations

SYSC Register

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

Bit 7 **CLK_ADJ:** Automatic Clock adjusts function
0: disable
1: enable (default)
Note that if the user selects the HIRC as the system clock, the CLK_ADJ bit must be set to "1" to adjust the clock frequency automatically.

Bit 6~0 Unimplemented, read as "0"

UCC Register

Bit	7	6	5	4	3	2	1	0
Name	Rctrl	SYSCLK	—	SUSP2	USBCKEN	—	EPS1	EPS0
R/W	R/W	R/W	—	R/W	R/W	—	R/W	R/W
POR	0	0	—	0	0	—	0	0

Bit 7 **Rctrl:** 7.5kΩ resistor between UDN and UBUS control bit
Described elsewhere

Bit 6 **SYSCLK:** Specify MCU oscillator frequency indication bit
0: 12MHz
1: 6MHz

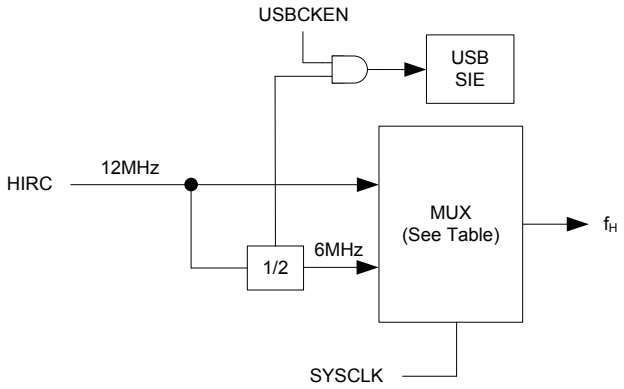
Bit 5 Unimplemented, read as "0"

Bit 4 **SUSP2:** Reduce power consumption in suspend mode control bit
Described elsewhere

Bit 3 **USBCKEN:** USB clock control bit
0: disable
1: enable

Bit 2 Unimplemented, read as "0"

Bit 1~0 **EPS1, EPS0:** Accessing endpoint FIFO selection
Described elsewhere



SYSCLK	f_H
0	12MHz
1	6MHz

High frequency system clock f_H configure

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 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. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins are free for use as normal I/O pins.

Internal 32kHz Oscillator – LIRC

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

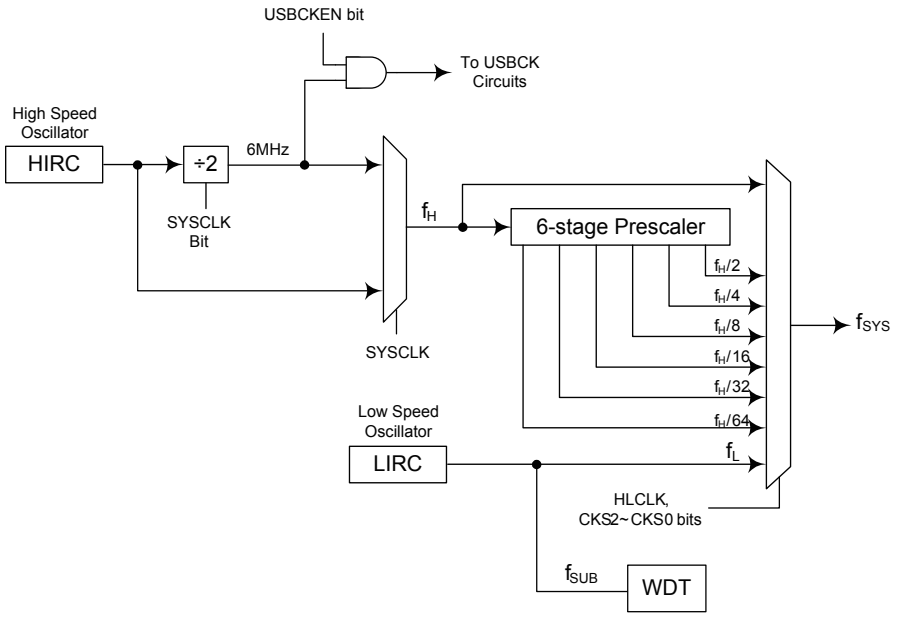
Operating Modes and System Clocks

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

System Clocks

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

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



System Clock Configuration

Note: When the system clock source f_{SYS} is switched to f_L from f_H, the high speed oscillation will stop to conserve the power. Thus there is no f_H~f_H/64 for peripheral circuit to use.

System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the NORMAL Mode and SLOW Mode. The remaining four modes, the SLEEP0, SLEEP1, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.

Operating Mode	Description		
	CPU	f_{SYS}	f_{SUB}
NORMAL Mode	on	f_H~f_H/64	on
SLOW Mode	on	f_L	on
IDLE0 Mode	off	off	on
IDLE1 Mode	off	on	on
SLEEP0 Mode	off	off	off
SLEEP1 Mode	off	off	on

NORMAL Mode

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

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from the low speed oscillator, LIRC. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW Mode, the f_H is off.

SLEEP0 Mode

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

SLEEP1 Mode

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

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU, the system oscillator will be stopped, the low frequency clock f_{SUB} will be on.

IDLE1 Mode

The IDLE1 Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU, the system oscillator will continue to run, and this system oscillator may be high speed or low speed system oscillator. In the IDLE1 Mode the low frequency clock f_{SUB} will be on.

Control Register

A single register, SMOD, is used for overall control of the internal clocks within the device.

SMOD Register

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

Bit 7~5 **CKS2~CKS0:** The system clock selection when HLCLK is "0"
000: f_L (f_{LIRC})
001: f_L (f_{LIRC})
010: $f_H/64$
011: $f_H/32$
100: $f_H/16$
101: $f_H/8$
110: $f_H/4$
111: $f_H/2$
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source, which can be LIRC, 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 **LTO:** Low speed system oscillator ready flag
0: Not ready
1: Ready
This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP0 Mode but after a wake-up has occurred, the flag will change to a high level after 2 clock cycles if the LIRC oscillator is used.

Bit 2 **HTO:** High speed system oscillator ready flag
0: Not ready
1: Ready
This is the high speed system oscillator ready flag which indicates when the high speed system oscillator is stable. This flag is cleared to "0" by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable. Therefore this flag will always be read as "1" by the application program after device power-on. The flag will be low when in the SLEEP or IDLE0 Mode but after a wake-up has occurred, the flag will change to a high level after 32 clock cycles if the HIRC oscillator is used.

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

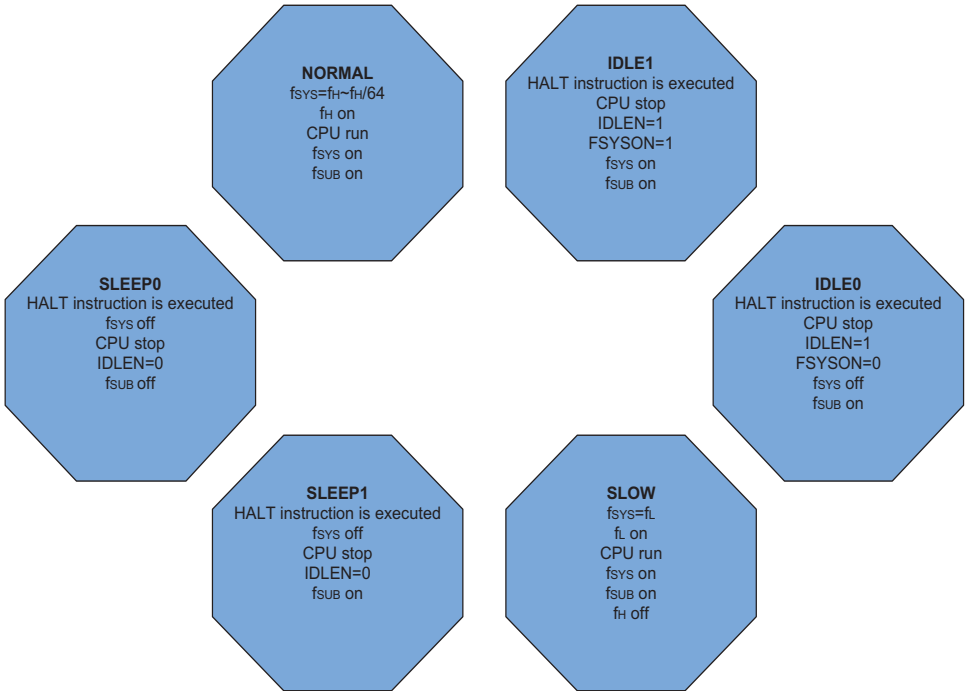
Bit 0 **HLCLK:** system clock selection
0: $f_H/2 \sim f_H/64$ or f_L
1: f_H
This bit is used to select if the f_H clock or the $f_H/2 \sim f_H/64$ or f_L clock is used as the system clock. When the bit is high the f_H clock will be selected and if low the $f_H/2 \sim f_H/64$ or f_L clock will be selected. When system clock switches from the f_H clock to the f_L clock and the f_H clock will be automatically switched off to conserve power.

Operating Mode Switching

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

In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SMOD register and FSYSON in the CTRL register.

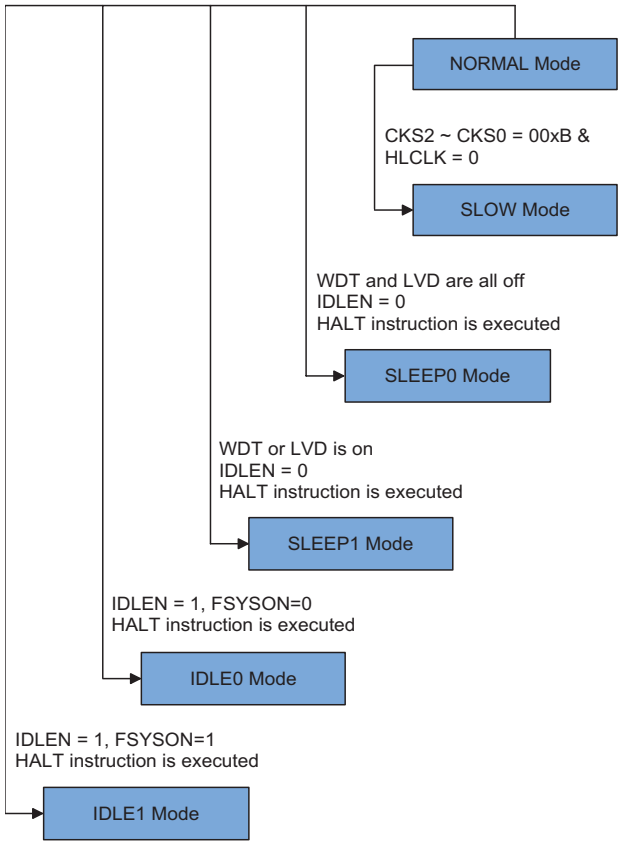
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~$f_H/64$ or f_L. If the clock is from the f_L, the high speed clock source will stop running to conserve power. When this happens it must be noted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



NORMAL Mode to SLOW Mode Switching

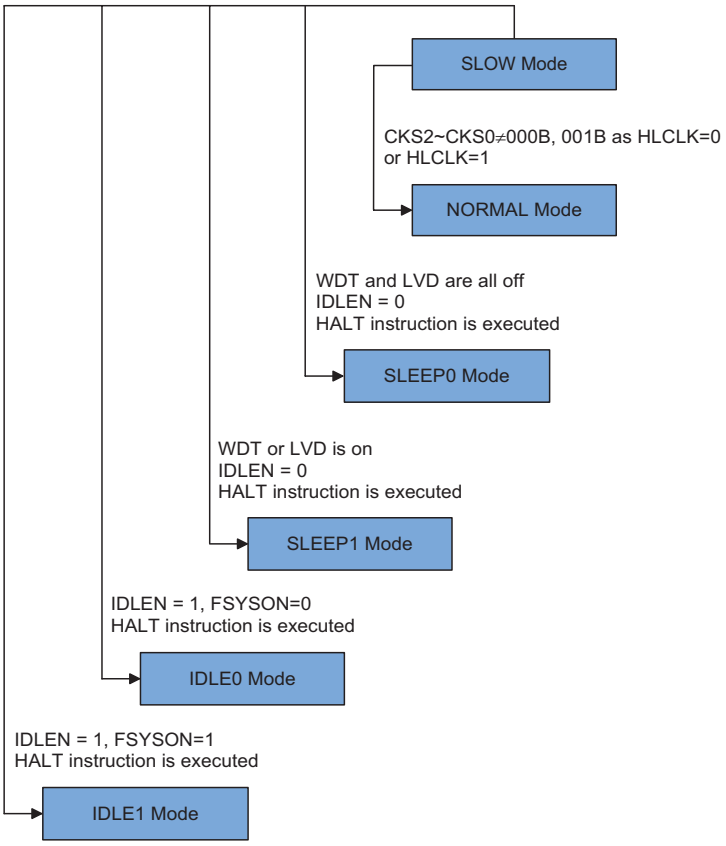
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to "0" and set the CKS2~CKS0 bits to "000" or "001" in the SMOD register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

The SLOW Mode is sourced from the LIRC oscillators and therefore requires these oscillators to be stable before full mode switching occurs. This is monitored using the LTO bit in the SMOD register.



SLOW Mode to NORMAL Mode Switching

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



Entering the SLEEP0 Mode

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

- The system clock and WDT 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 WDT will be cleared and stopped.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the SLEEP1 Mode

There is only one way for the device to enter the SLEEP1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0" and the WDT or LVD on. When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the "HALT" instruction, but the WDT or LVD will remain with the clock source coming from the f_{SUB} clock.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT is enabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

There is only one way for the device to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in CTRL register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

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

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 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be 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 is enabled.

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

Wake-up

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

- An external or USB reset
- An external falling edge on Ports
- A system interrupt
- A WDT overflow

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

Programming Considerations

The high speed and low speed oscillators both use the same SST counter. For example, if the system is woken up from the SLEEP0 Mode and the HIRC oscillator needs to start-up from an off state. The LIRC oscillator uses the SST counter after HIRC oscillator has finished its SST period.

- If the device is woken up from the SLEEP0 Mode to the NORMAL Mode, the high speed system oscillator needs an SST period. The device will execute first instruction after HTO is "1".
- There are peripheral functions, such as WDT, TMs and SIM, for which the f_{SYS} is used. If the system clock source is switched from f_H to f_L, the clock source to the peripheral functions mentioned above will change accordingly.
- The on/off condition of f_L depends upon whether the WDT is enabled or disabled as the WDT clock source is generated from f_L.

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_{SUB}, which is sourced from the LIRC oscillator. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register. The LIRC internal oscillator has an approximate period of 32kHz at a supply voltage of 5V.

However, it should be noted that this specified internal clock period can vary with V_{DD}, temperature and process variations. The WDT function is allowed to enable or disable by setting the WDTC register data.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. The WRF software reset flag will be indicated in the CTRL register.

WDTC Register

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

Bit 7~3 **WE4~WE0:** WDT function software control
10101: disable
01010: enable
Other: reset MCU
When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the CTRL register will be set to 1 to indicate the reset source.

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

CTRL Register

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

"x" unknown

Bit 7 **FSYSON:** f_{SYS} Control in IDLE Mode
0: disable
1: enable

Bit 6~3 Unimplemented, read as "0"

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

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

Bit 0 **WRF:** WDT Control register software reset flag
0: not occur
1: occurred
This bit is set 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 instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer additional enable/disable and reset control of the Watchdog Timer.

WDT Enable/Disabled using the WDT Control Register

The WDT is enabled/disabled using the WDT control register, the WE4~WE0 values can determine which mode the WDT operates in. The WDT will be disabled when the WE4~WE0 bits are set to a value of 10101B. The WDT function will be enabled if the WE4~WE0 bit value is equal to 01010B. If the WE4~WE0 bits are set to any other values other than 01010B and 10101B, it will reset the device after 2~3 LIRC clock cycles. After power on these bits will have the value of 01010B.

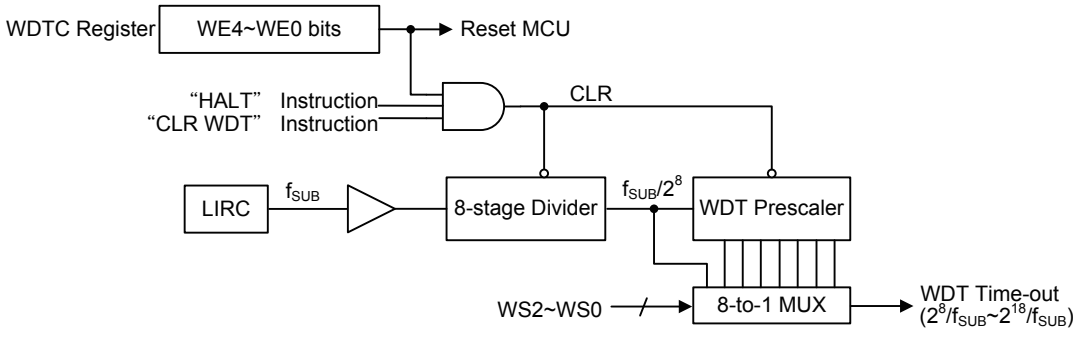
WDT Configuration Option	WE4~WE0 Bits	WDT Function
Controlled by WDT Control Register	10101B	Disable
	01010B	Enable
	Any other value	Reset MCU

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 bit filed, the second is using the Watchdog Timer software clear instructions and the

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

The maximum time out period is when the 2^{18} division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 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. A hardware reset will of course be automatically implemented after the device is powered on, however there is a number of other hardware and software reset sources that can be implemented dynamically when the device is running.

Reset Overview

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 instructions commence execution. 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.

The devices provide several reset sources to generate the internal reset signal, providing extended MCU protection. The different types of resets are listed in the accompanying table.

Reset Name	Abbreviation	Indication Bit	Register	Notes
Power-On Reset	POR	—	—	Auto generated at power on
Reset Pin	$\overline{\text{RES}}$	—	—	Hardware Reset
Low-Voltage Reset	LVR	LRF	CTRL	Low VDD voltage
Watchdog Reset	WDT	TO	STATUS	
WDTC Register Setting Software Reset	—	WRF	CTRL	Write to WDTC register
LVRC Register Setting Software Reset	—	LRF	CTRL	Write to LVRC register

Reset Source Summary

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

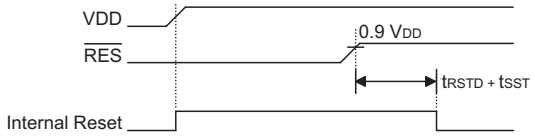
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{RES}$ reset is implemented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

Reset Functions

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

Power-on Reset

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



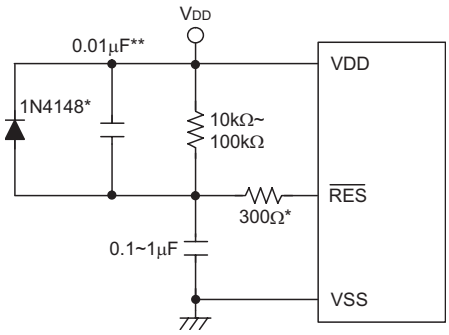
Power-On Reset Timing Chart

$\overline{RES}$ Pin

Although the microcontroller has an internal RC reset function, if the VDD power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{RES}$ pin, whose additional time delay will ensure that the $\overline{RES}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{RES}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an 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 $\overline{RES}$ pin and a capacitor connected between VSS and the $\overline{RES}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{RES}$ pin should be kept as short as possible to minimise any stray noise interference.

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



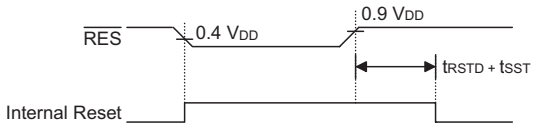
External $\overline{\text{RES}}$ Circuit

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.

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

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



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

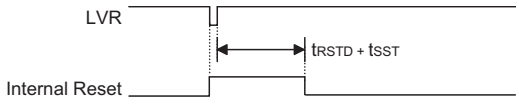
Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provide an MCU reset should the value fall below a certain predefined level.

LVR Operation

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 CTRL 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 A.C. 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 are changed to some certain values by the environmental noise or software setting, the LVR will reset the device after 2~3 LIRC clock cycles. When this happens, the LRF bit in the CTRL register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the power down mode.



Low Voltage Reset Timing Chart

• **LVRC Register**

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

Bit 7~0 **LVS7~LVS0:** LVR voltage select
01010101: 2.1V (default)
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Any other value: Generates MCU reset – LVRC register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage values above, an MCU reset will be generated. In this situation the register contents will remain the same after such a reset occurs.

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

• **CTRL Register**

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

"x" unknown

Bit 7 **FSYSON:** f_{SYS} Control in IDLE Mode
Described elsewhere

Bit 6~3 Unimplemented, read as "0"

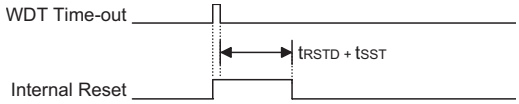
Bit 2 **LVRF:** LVR function reset flag
0: not occur
1: occurred
This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

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

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

Watchdog Time-out Reset during Normal Operation

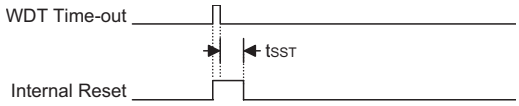
The Watchdog time-out Reset during normal operation is the same as a hardware $\overline{RES}$ pin reset except that the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during Sleep Timing Chart

WDTC Register Software Reset

A WDTC software reset will be generated when a value other than "10101" or "01010", exist in the highest five bits of the WDTC register. The WRF bit in the CTRL register will be set high when this occurs, thus indicating the generation of a WDTC software reset.

- **WDTC Register**

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

Bit 7~3 **WE4~WE0:** WDT function software control
10101: disable
01010: enable (default)
Other: reset MCU
When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the CTRL register will be set to 1 to indicate the reset source.

Bit 2~0 **WS2~WS0:** WDT time-out period selection
Described elsewhere

Software Reset Control

The devices provide the ability to control the MCU reset function enable or not by software option. This function is managed by the FSWRST bit in the FRCR register.

- **FRCR Register**

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

Bit 7~5 Unimplemented, read as "0"

Bit 4 **FSWRST:** Software Reset MCU control bit
0: MCU reset enable
1: MCU reset disable (except power on reset)
This control bit is used to setup the reset action of MCU when there is a reset signal detected by the USB SIE. If this bit is set to "0", the MCU reset will happen and if this bit is set to "1", then the reset signal will be bypassed, the MCU will not be reset. Note that the power on reset will not be controlled by the FSWRST bit.

Bit 3~1 unimplemented, read as "0"

Bit 0 **CLWB:** Flash Program memory Write buffer clear control bit
Described elsewhere

Reset Initial Conditions

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

TO	PDF	Reset Conditions
0	0	Power-on reset
u	u	$\overline{\text{RES}}$, 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

Note: "u" stands for unchanged

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

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

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

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)*	USB-reset (Normal)	USB-reset (HALT)
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
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	---- xxxx	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu	--uu uuuu	--uu uuuu
BP	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
SMOD	000- 0011	000- 0011	000- 0011	000- 0011	uuu- uuuu	000- 0011	000- 0011
INTEG	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu	---- 0000	---- 0000
LVDC	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu	--00 -000	--00 -000
WDTC	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu	0101 0011	0101 0011
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu	-000 -000	-000 -000
INTC2	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu	-000 -000	-000 -000
MFI0	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
PAPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
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
PXWU	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000
PXPU	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 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
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
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PE	---- --11	---- --11	---- --11	---- --11	---- --uu	---- --11	---- --11
PEC	---- --11	---- --11	---- --11	---- --11	---- --uu	---- --11	---- --11
I2CTOC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
SIMC0	1110 000-	1110 000-	1110 000-	1110 000-	uuuu uuu-	1110 000-	1110 000-
SIMC1	1000 0001	1000 0001	1000 0001	1000 0001	uuuu uuuu	1000 0001	1000 0001
SIMD	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
SIMA/ SIMC2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
SBSC	0-00 ----	0-00 ----	0-00 ----	0-00 ----	u-uu ----	0-00 ----	0-00 ----
TMPC	--01 --01	--01 --01	--01 --01	--01 --01	--uu --uu	--01 --01	--01 --01
TM0C0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TM0C1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TM0DL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TM0DH	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
TM0AL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TM0AH	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*	USB-reset (Normal)	USB-reset (HALT)
TM1C0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TM1C1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TM1DL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TM1DH	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
TM1AL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TM1AH	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
FRCR	---0 ---0	---0 ---0	---0 ---0	---0 ---0	---u ---u	---0 ---0	---0 ---0
FCR	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
FARH	---- xxxx	---- xxxx	---- xxxx	---- xxxx	---- uuuu	---- xxxx	---- xxxx
FD0L	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
FD1L	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
FD2L	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
FD3L	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
USB_STAT	11xx 000-	uuxx 00u-	uuxx 00u-	uuxx 00u-	uuxx 00u-	uuxx 00u-	uuxx 00u-
UINT	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu	-000 -000	-000 -000
USC	10-0 x0xx	uu-u xuxx	uu-u xuxx	uu-u xuxx	uu-u xuxx	uu-u 0100	uu-u 0100
UCC	00-0 0-xx	uu-u u-xx	uu-u u-xx	uu-u u-xx	uu-u u-uu	uu-u u-00	uu-u u-00
AWR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	0000 0000	0000 0000
STL	-xxx -xxx	-xxx -xxx	-xxx -xxx	-xxx -xxx	-xxx -xxx	-000 -000	-000 -000
SIES	xx-x xxxx	xx-x xxxx	xx-x xxxx	xx-x xxxx	xx-x xxxx	00-0 0000	00-0 0000
MISC	xxx- -xxx	xxx- -xxx	xxx- -xxx	xxx- -xxx	xxx- -xxx	000- -000	000- -000
UFEN	-00- -000	-uu- -uuu	-00- -000	-00- -000	-uu- -uuu	-00- -000	-00- -000
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
URDCT	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	0000 0000	0000 0000
PWMC	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu	0--- -000	0--- -000
PWM0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWM1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWM2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CTRL	0--- -x00	0--- -x00	0--- -x00	0--- -x00	u--- -xuu	0--- -x00	0--- -x00
LVRC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu	0101 0101	0101 0101
SYSC	1--- ----	1--- ----	1--- ----	1--- ----	u--- ----	1--- ----	1--- ----

Note: "u" stands for unchanged
"x" stands for unknown
"-" stands for unimplemented

Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA~PE. 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.

I/O Register List

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PXWU	—	PELWU	PDHWU	PDLWU	PCHWU	PCLWU	PBHWU	PBLWU
PXPU	—	PELPU	PDHPU	PDLPU	PCHPU	PCLPU	PBHPU	PBLPU
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PE	—	—	—	—	—	—	PE1	PE0
PEC	—	—	—	—	—	—	PEC1	PEC0

Pull-high Resistors

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

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 Control
0: Disable
1: Enable

PXPU Register

Bit	7	6	5	4	3	2	1	0
Name	—	PELWU	PDHWU	PDLWU	PCHWU	PCLWU	PBHWU	PBLWU
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 **PELPU:** PE1~PE0 pins Pull-High control
0: Disable
1: Enable

Bit 5 **PDHPU:** PD7~PD4 pins Pull-High control
0: Disable
1: Enable

Bit 4 **PDLPU:** PD3~PD0 pins Pull-High control
0: Disable
1: Enable

Bit 3 **PCHPU:** PC7~PC4 pins Pull-High control
0: Disable
1: Enable

Bit 2 **PCLPU:** PC3~PC0 pins Pull-High control
0: Disable
1: Enable

Bit 1 **PBHPU:** PB7~PB4 pins Pull-High control
0: Disable
1: Enable

Bit 0 **PBLPU:** PB3~PB0 pins Pull-High control
0: Disable
1: Enable

Port 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 ~ Port E 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 ~ Port E can be selected respectively to have this wake-up feature using the PAWU and PXWU registers.

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 Control
0: Disable
1: Enable

PXWU Register

Bit	7	6	5	4	3	2	1	0
Name	—	PELPU	PDHPU	PDLPU	PCHPU	PCLPU	PBHPU	PBLPU
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 **PELWU:** PE1~PE0 pins Wake-up control
0: Disable
1: Enable

Bit 5 **PDHWU:** PD7~PD4 pins Wake-up control
0: Disable
1: Enable

Bit 4 **PDLWU:** PD3~PD0 pins Wake-up control
0: Disable
1: Enable

Bit 3 **PCHWU:** PC7~PC4 pins Wake-up control
0: Disable
1: Enable

Bit 2 **PCLWU:** PC3~PC0 pins Wake-up control
0: Disable
1: Enable

Bit 1 **PBHWU:** PB7~PB4 pins Wake-up control
0: Disable
1: Enable

Bit 0 **PBLWU:** PB3~PB0 pins Wake-up control
0: Disable
1: Enable

Note: I/O port slew rate always 200ns in HT68FB240. If the SIM function is enabled, SPI/I^2C function pin slew rate is kept 0 ns.

I/O Port Control Registers

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

PAC Register

Bit	7	6	5	4	3	2	1	0
Name	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PAC7~PAC0:** Port A bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

PBC Register

Bit	7	6	5	4	3	2	1	0
Name	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PBC7~PBC0:** Port B bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

PCC Register

Bit	7	6	5	4	3	2	1	0
Name	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PCC7~PCC0:** Port C bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

PDC Register

Bit	7	6	5	4	3	2	1	0
Name	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PDC7~PDC0:** Port D bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

PEC Register

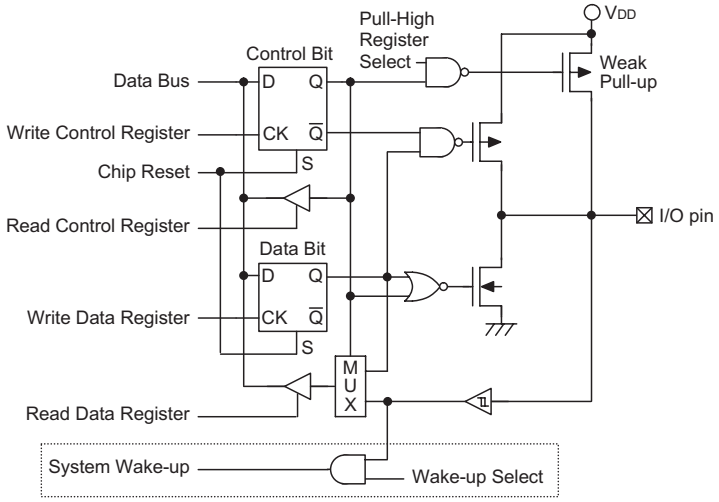
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PEC1	PEC0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	1	1

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PEC1~PEC0:** Port E bit 1 ~ bit 0 Input/Output Control
0: Output
1: Input

I/O Pin Structures

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



Generic Input/Output Structure

Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers, PAC~PEC, 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, PA~PE, 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.

All Ports provide the wake-up function which can be set by individual pin in the Port A while it has to be set by nibble pins in the Port B, Port C, Port D and Port E. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a voltage level transition on any of the Port pins. Single or multiple pins on Ports can be setup to have this function.

Timer Modules – TM

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

Introduction

The device contains two TMs having a reference name of TM0 and TM1. The TMs can be categorised as Compact Type TM. The main features of CTMs are summarised in the accompanying table.

Function	CTM
Timer/Counter	√
Compare Match Output	√
PWM Channels	1
PWM Alignment	Edge
PWM Adjustment Period & Duty	Duty or Period

This chip contains a specific number of Compact Type TM unit which is shown in the table together with their individual reference names, TM0~TM1.

Device	TM0	TM1
HT68FB240	10-bit CTM	10-bit CTM

TM Name/Type Reference

TM Operation

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

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the TnCK2~TnCK0 bits in the TM control registers. The clock source can be a ratio of either the system clock f_{SYS} or the internal high clock f_H, the f_{LIRC} clock source or the external TCKn pin. Note that setting these bits to the value 101 will select a reserved clock input, in effect disconnecting the TM clock source. The TCKn pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

The Compact Type TMs have two internal interrupts, 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 TM input pin, with the label TCKn. The TM input pin is essentially a clock source for the TM and is selected using the TnCK2~TnCK0 bits in the TMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. This external TM input pin is shared with other functions but will be connected to the internal TM if selected using the TnCK2~TnCK0 bits. The TM input pin can be chosen to have either a rising or falling active edge.

The TMs each have more output pins with the label TPn, TPnB. 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 TPn or TPnB output pin is also the pin where the TM generates the PWM output waveform. As the TM output pins are pin-shared with other function, the TM output function must first be setup using registers. A single bit in one of the registers determines if its associated pin is to be used as an external TM output pin or if it is to have another function. The number of output pins for each TM type and device is different, the details are provided in the accompanying table.

This allows the TM to generate a complimentary output pair, selected using the I/O register data bits.

Device	CTM	Register
HT68FB240	TP0, TP0B TP1, TP1B	TMPC

TM Output Pins

TM Input/Output Pin Control Registers

Selecting to have a TM input/output or whether to retain its other shared functions is implemented using one register with a single bit in each register corresponding to a TM input/output pin. Setting the bit high will setup the corresponding pin as a TM input/output if reset to zero the pin will retain its original other functions.

TMPC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	T1CP1	T1CP0	—	—	T0CP1	T0CP0
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	1	—	—	0	1

Bit 7~6 Unimplemented, read as "0"

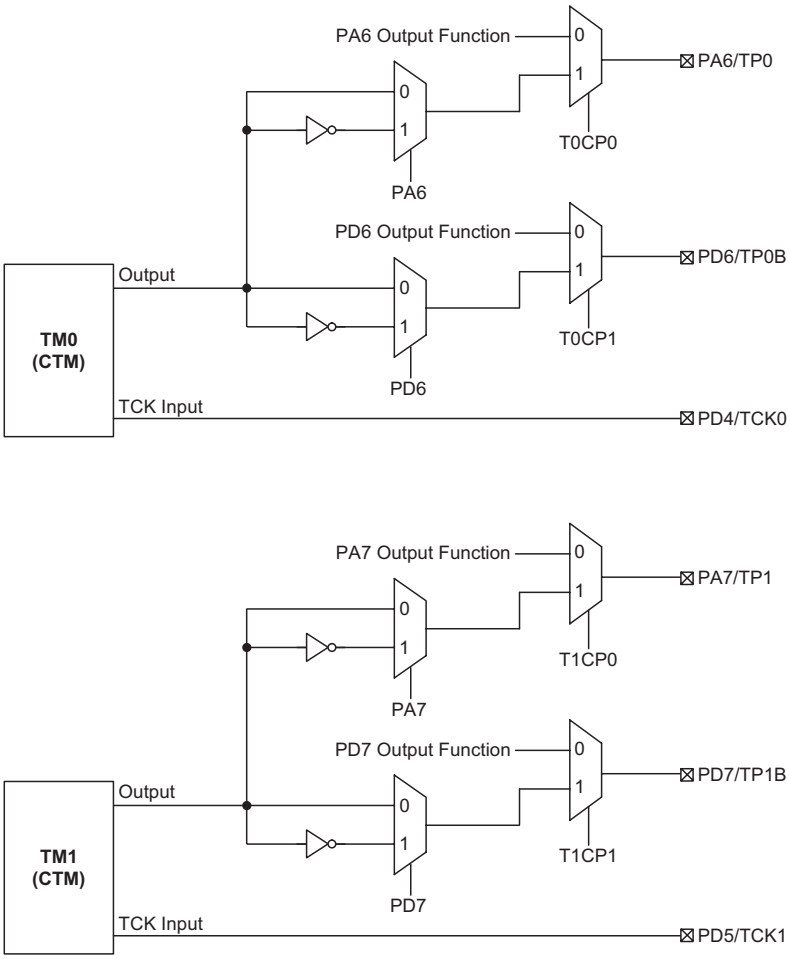
Bit 5 **T1CP1:** TP1B pin control
0: Disable
1: Enable

Bit 4 **T1CP0:** TP1 pin control
0: Disable
1: Enable

Bit 3~2 Unimplemented, read as "0"

Bit 1 **T0CP1:** TP0B pin control
0: Disable
1: Enable

Bit 0 **T0CP:** TP0 pin control
0: Disable
1: Enable

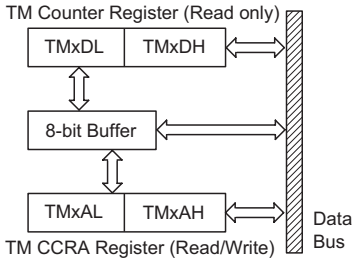


TM0 & TM1 Function Pin Control Block Diagram

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA registers, being 10-bit, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed.

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



The following steps show the read and write procedures:

- Writing Data to CCRA
  - Step 1. Write data to Low Byte TMxAL
    – Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte TMxAH
    – Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA
  - Step 1. Read data from the High Byte TMxDH or TMxAH
    – 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 TMxDL or TMxAL
    – This step reads data from the 8-bit buffer.

Compact Type TM – CTM

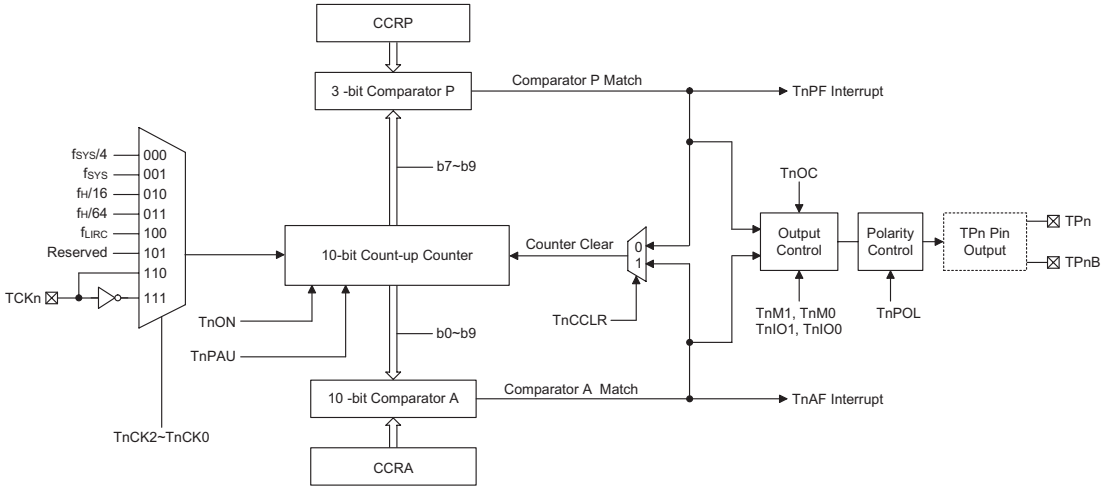
Although the simplest form of the three TM types, the Compact TM type still contains three operating modes, which are Compare Match Output, Timer/Event Counter and PWM Output modes. The Compact TM can also be controlled with an external input pin and can drive two external output pins. These two external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

Name	TM No.	TM Input Pin	TM Output Pin
10-bit CTM	0, 1	TCK0, TCK1	TP0, TP0B TP1, TP1B

Compact TM Operation

At its core is a 10-bit count-up counter which is driven by a user selectable internal clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is three bits wide whose value is compared the highest three bits in the counter while the CCRA is the ten bits and therefore compares 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 TnON 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 Block Digram (n=0, 1)

Compact Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMnC0	TnPAU	TnCK2	TnCK1	TnCK0	TnON	TnRP2	TnRP1	TnRP0
TMnC1	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnDPX	TnCCLR
TMnDL	D7	D6	D5	D4	D3	D2	D1	D0
TMnDH	—	—	—	—	—	—	D9	D8
TMnAL	D7	D6	D5	D4	D3	D2	D1	D0
TMnAH	—	—	—	—	—	—	D9	D8

Compact TM Register List (n=0, 1)

TMnC0 Register (n=0, 1)

Bit	7	6	5	4	3	2	1	0
Name	TnPAU	TnCK2	TnCK1	TnCK0	TnON	TnRP2	TnRP1	TnRP0
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 **TnPAU:** TMn 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 TM 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 **TnCK2~TnCK0:** Select TMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{LIRC}
101: Undefined
110: TCKn rising edge clock
111: TCKn falling edge clock

These three bits are used to select the clock source for the TM. 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_{LIRC} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **TnON:** TMn Counter On/Off Control
0: Off
1: On

This bit controls the overall on/off function of the TM. Setting the bit high enables the counter to run, clearing the bit disables the TM. Clearing this bit to zero will stop the counter from counting and turn off the TM 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. If the TM is in the Compare Match Output Mode then the TM

output pin will be reset to its initial condition, as specified by the TnOC bit, when the TnON bit changes from low to high.

Bit 2~0 **TnRP2~TnRP0:** TMn CCRP 3-bit register, compared with the TMn Counter bit 9~bit 7 Comparator P Match Period
000: 1024 TMn clocks
001: 128 TMn clocks
010: 256 TMn clocks
011: 384 TMn clocks
100: 512 TMn clocks
101: 640 TMn clocks
110: 768 TMn clocks
111: 896 TMn 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 TnCCLR bit is set to zero. Setting the TnCCLR 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.

TMnC1 Register (n=0, 1)

Bit	7	6	5	4	3	2	1	0
Name	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnDPX	TnCCLR
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 **TnM1, TnM0:** Select TMn 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 TM. To ensure reliable operation the TM should be switched off before any changes are made to the TnM1 and TnM0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5~4 **TnIO1, TnIO0:** Select TPn, TPnB output function
Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output
PWM 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 TM output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the TM is running.

In the Compare Match Output Mode, the TnIO1 and TnIO0 bits determine how the TM 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 TM output pin should be setup using the TnOC bit in the TMnC1 register. Note that the output level requested by the TnIO1 and TnIO0 bits must be different from the initial value setup using the TnOC bit otherwise no change will occur on the TM output pin when a compare match occurs. After the TM output pin changes state, it can be reset to its initial level by changing the level of the TnON bit from low to high.

In the PWM Mode, the TnIO1 and TnIO0 bits determine how the TM 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 TnIO1 and TnIO0 bits only after the TMn has been switched off. Unpredictable PWM outputs will occur if the TnIO1 and TnIO0 bits are changed when the TM is running.

Bit 3 **TnOC:** TPn, TPnB Output control bit

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

PWM Mode
0: Active low
1: Active high

This is the output control bit for the TM output pin. Its operation depends upon whether TM is being used in the Compare Match Output Mode or in the PWM Mode. It has no effect if the TM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the TM 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 **TnPOL:** TPn, TPnB Output polarity Control
0: Non-invert
1: Invert

This bit controls the polarity of the TPn, TPnB output pin. When the bit is set high the TM 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 **TnDPX:** TMn PWM period/duty Control
0: CCRP - period; CCRA - duty
1: CCRP - duty; CCRA - period

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

Bit 0 **TnCCLR:** Select TMn Counter clear condition
0: TMn Comparatror P match
1: TMn Comparatror A match

This bit is used to select the method which clears the counter. Remember that the Compact TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the TnCCLR 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 TnCCLR bit is not used in the PWM Mode.

TMnDL Register (n=0, 1)

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

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

TMnDH Register (n=0, 1)

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

Bit 7~2 Unimplemented, read as "0"

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

TMnAL Register (n=0, 1)

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

Bit 7~0 **D7~D0:** TMn CCRA Low Byte Register bit 7 ~ bit 0
TMn 10-bit CCRA bit 7 ~ bit 0

TMnAH Register (n=0, 1)

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

Bit 7~2 Unimplemented, read as "0"

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

Compact Type TM Operating Modes

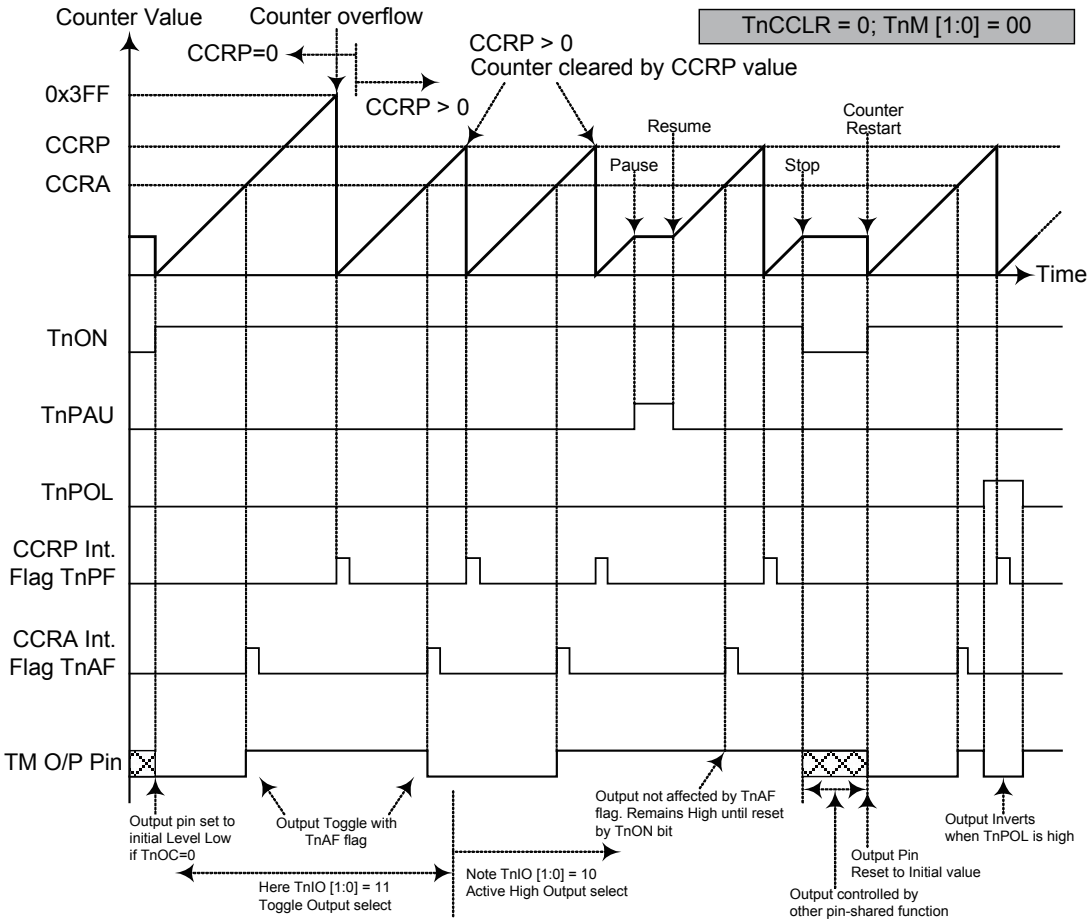
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Mode or Timer/Counter Mode. The operating mode is selected using the TnM1 and TnM0 bits in the TMnC1 register.

Compare Match Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 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 TnCCLR 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 TnAF and TnPF interrupt request flags for the Comparator A and Comparator P respectively, will both be generated.

If the TnCCLR bit in the TMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the TnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when TnCCLR is high no TnPF 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 TnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the TM output pin will change state. The TM output pin condition however only changes state when a TnAF interrupt request flag is generated after a compare match occurs from Comparator A. The TnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the TM output pin. The way in which the TM output pin changes state are determined by the condition of the TnIO1 and TnIO0 bits in the TMnC1 register. The TM output pin can be selected using the TnIO1 and TnIO0 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 TM output pin, which is setup after the TnON bit changes from low to high, is setup using the TnOC bit. Note that if the TnIO1 and TnIO0 bits are zero then no pin change will take place.

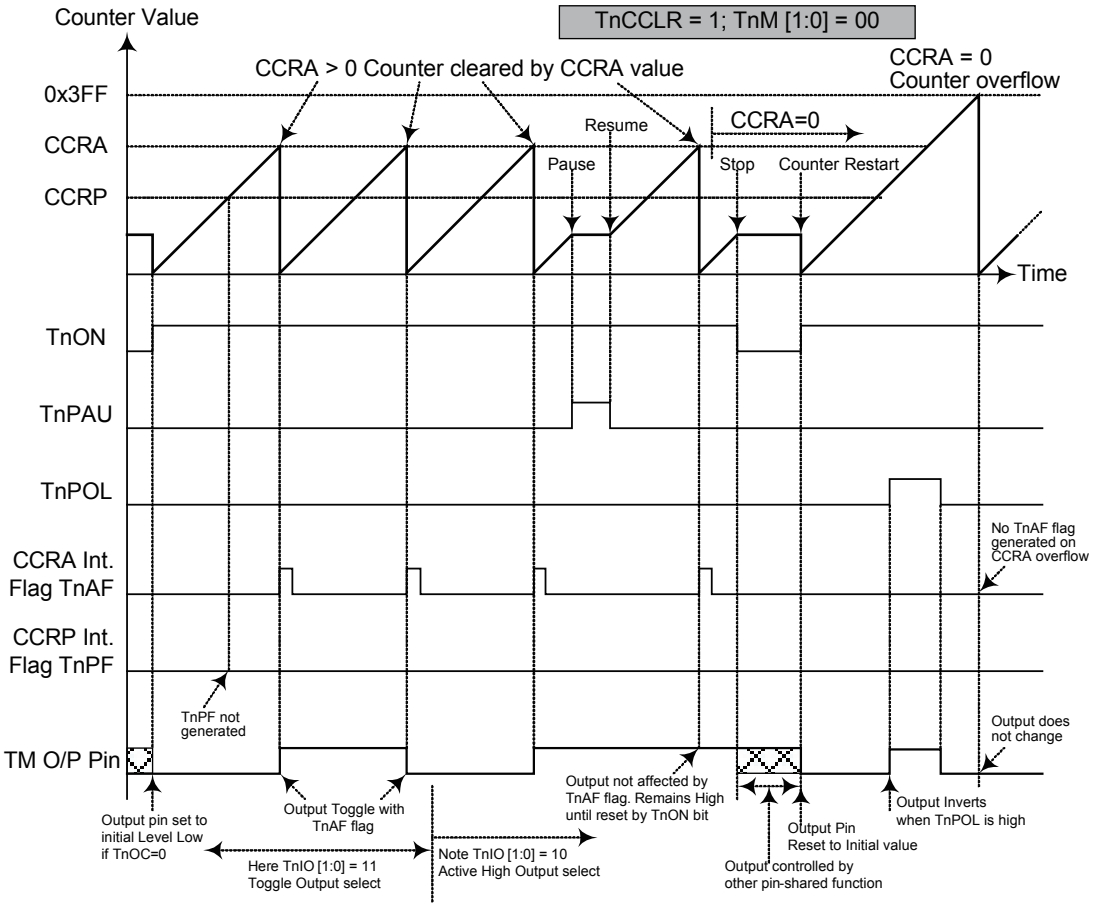


Compare Match Output Mode – TnCCLR = 0 (n=0, 1)

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

2. The TM output pin is controlled only by the TnAF flag

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



Compare Match Output Mode – TnCCLR = 1 (n=0, 1)

Note: 1. With TnCCLR=1, a Comparator A match will clear the counter

2. The TM output pin is controlled only by the TnAF flag

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

4. The TnPF flag is not generated when TnCCLR=1

Timer/Counter Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 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 TM 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 TM 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 TnM1 and TnM0 in the TMnC1 register should be set to "10" respectively. The PWM function within the TM 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 TM 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 TnCCLR 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 TnDPX bit in the TMnC1 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 TnOC bit in the TMnC1 register is used to select the required polarity of the PWM waveform while the two TnIO1 and TnIO0 bits are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The TnPOL bit is used to reverse the polarity of the PWM output waveform.

10-bit CTM, PWM Mode, Edge-aligned Mode, TnDPX=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, TM clock source select $f_{SYS}/4$, CCRP = 100b and CCRA = 128,

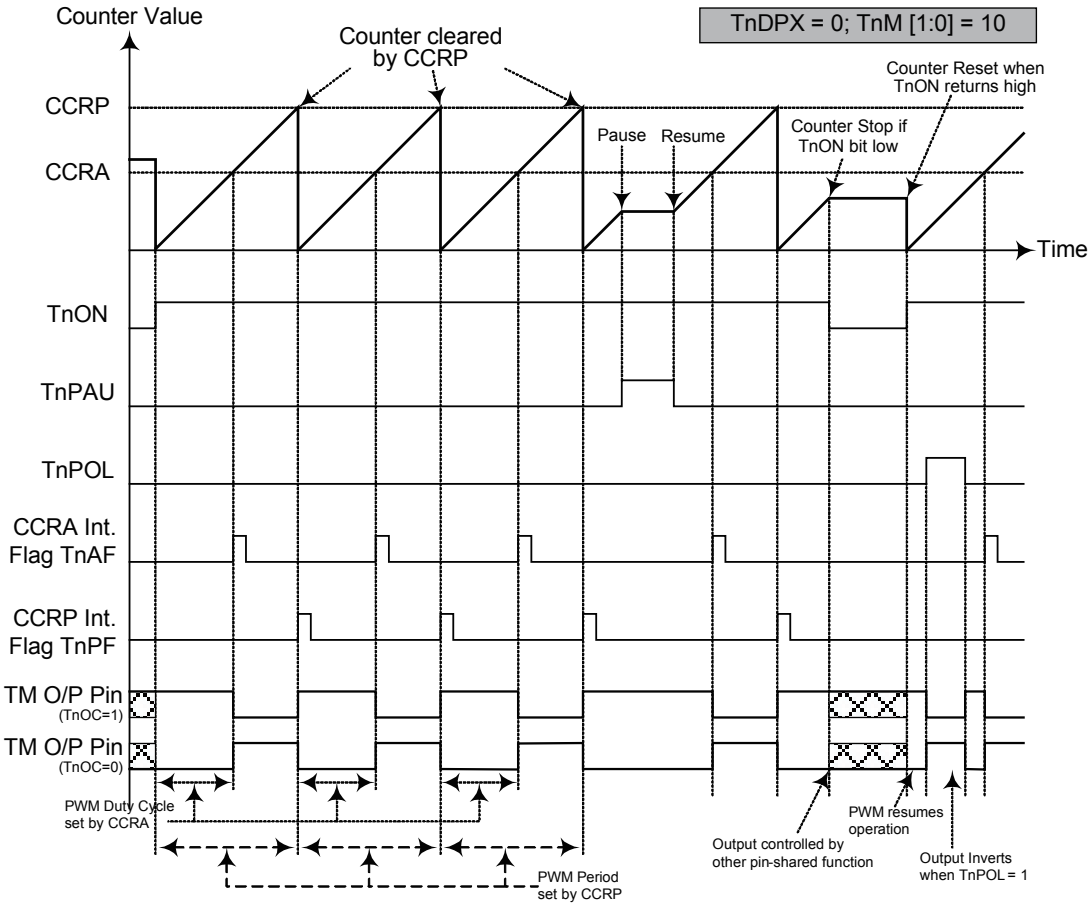
The CTM PWM output frequency = $(f_{SYS}/4)/512 = f_{SYS}/2048$ = 7.8125kHz, duty = 128/512 = 25%.

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%.

10-bit CTM, PWM Mode, Edge-aligned Mode, TnDPX=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 TM clock while the PWM duty cycle is defined by the CCRP register value.



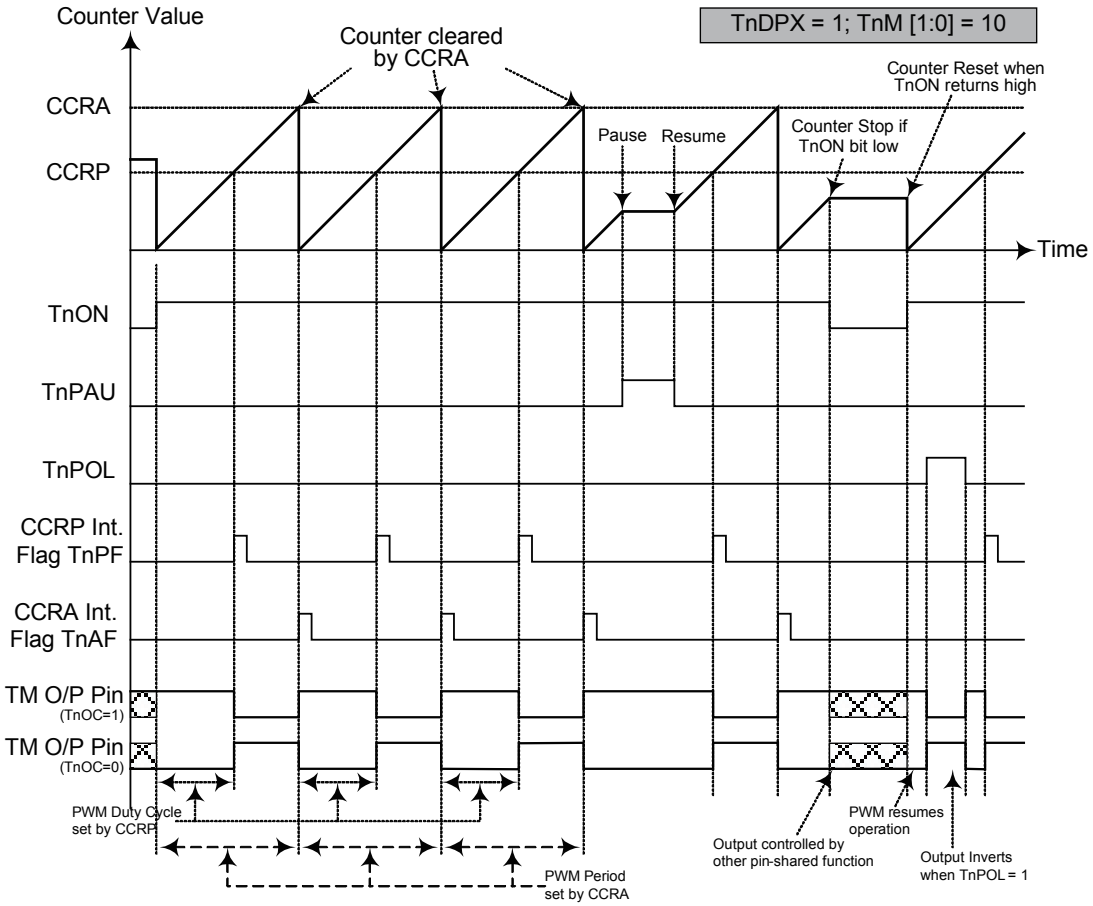
PWM Mode – TnDPX = 0 (n=0, 1)

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

2. A counter clear sets the PWM Period

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

4. The TnCCLR bit has no influence on PWM operation



PWM Mode – TnDPX = 1 (n=0, 1)

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

2. A counter clear sets the PWM Period

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

4. The TnCCLR bit has no influence on PWM operation

Serial Interface Module – SIM

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

There is one control register associated with the serial interface control, namely SBSC. This is used to enable the SIM WCOL function and I²C debounce selection.

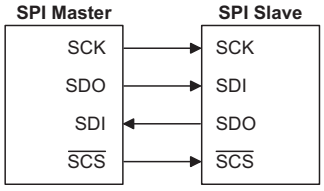
SPI Interface

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

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

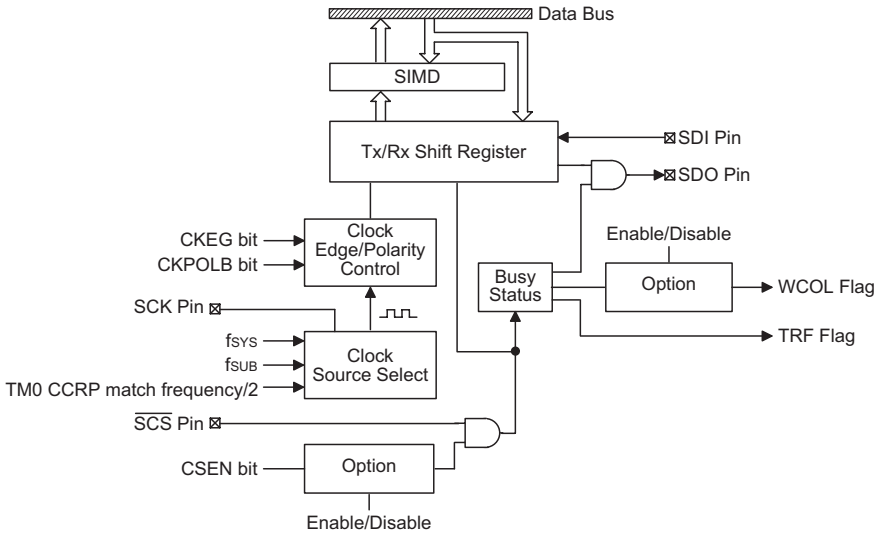
SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, SCK is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I²C function pins, the SPI interface must first be enabled by selecting the SIM enable configuration option and setting the correct bits in the SIMC0 and SIMC2 registers. After the SPI option has been selected, it can also be additionally disabled or enabled using the SIMEN bit in the SIMC0 register.



SPI Master/Slave Connection

Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ pin only one slave device can be utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to 1 to enable $\overline{SCS}$ pin function, set CSEN bit to 0 the $\overline{SCS}$ pin will be floating state.



SPI Block Diagram

The SPI function in this device offers the following features:

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

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

SPI Registers

There are four internal registers which control the overall operation of the SPI interface. These are the SIMD data register and three registers SIMC0, SIMC2 and SBSC. Note that the SIMC1 register is only used by the I²C interface.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
SBSC	SIM_WCOL	—	I2CDB1	I2CDB0	—	—	—	—

SIM Registers List

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

The SIM_WCOL bit in the SBSC register is used to control the SPI WCOL function.

SIMD Register

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

"x" unknown

There are also three control registers for the SPI interface, SIMC0, SIMC2 and SBSC. Note that the SIMC2 register also has the name SIMA which is used by the I²C function. The SIMC1 register is not used by the SPI function, only by the I²C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Although not connected with the SPI function, the SIMC0 register is also used to control the Peripheral Clock Prescaler. Register SIMC2 is used for other control functions such as LSB/MSB selection, write collision flag etc.

SIMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	1	1	1	0	0	0	0	—

Bit 7~5 **SIM2~SIM0:** SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{LIRC}
100: SPI master mode; SPI clock is TM0 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: I/O mode

These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from the TM0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **PCKEN:** PCK Output Pin Control
0: Disable
1: Enable

Bit 3~2 **PCKP1, PCKP0:** Select PCK output pin frequency
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/8$
11: TM0 CCRP match frequency/2

Bit 1 **SIMEN:** SIM Control
0: Disable
1: Enable

The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will be in a floating condition and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I²C interface via the

SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 Unimplemented, read as "0"

SIMC2 Register

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

Bit 7~6 **D7, D6:** Undefined bit

This bit can be read or written by user software program.

Bit 5 **CKPOLB:** Determines the base condition of the clock line

0: the SCK line will be high when the clock is inactive

1: the SCK line will be low when the clock is inactive

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

Bit 4 **CKEG:** Determines SPI SCK active clock edge type

CKPOLB=0

0: SCK is high base level and data capture at SCK rising edge

1: SCK is high base level and data capture at SCK falling edge

CKPOLB=1

0: SCK is low base level and data capture at SCK falling edge

1: SCK is low base level and data capture at SCK rising edge

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

Bit 3 **MLS:** SPI Data shift order

0: LSB

1: MSB

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

Bit 2 **CSEN:** SPI $\overline{SCS}$ pin Control

0: Disable

1: Enable

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

Note that using the CSEN bit can be disabled or enabled via configuration option.

Bit 1 **WCOL:** SPI Write Collision flag

0: No collision

1: Collision

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

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

SBSC Register

Bit	7	6	5	4	3	2	1	0
Name	SIM_WCOL	—	I2CDB1	I2CDB0	—	—	—	—
R/W	R/W	—	R/W	R/W	—	—	—	—
POR	0	—	0	0	—	—	—	—

Bit 7 **SIM_WCOL:** SIM WCOL control bit
0: Disable
1: Enable

Bit 6 Unimplemented, read as "0"

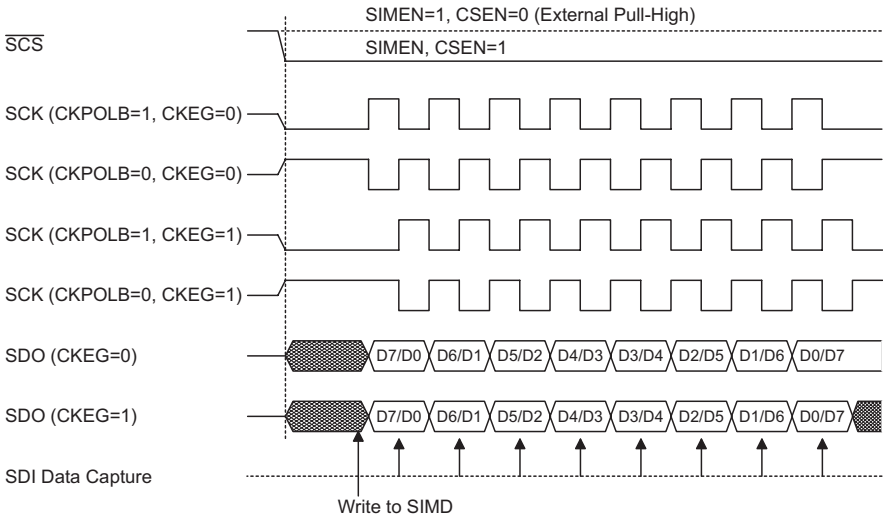
Bit 5~4 **I2CDB1~I2CDB0:** I^2C debounce selection bits
Related to I^2C function, described elsewhere

Bit 3~0 Unimplemented, read as "0"

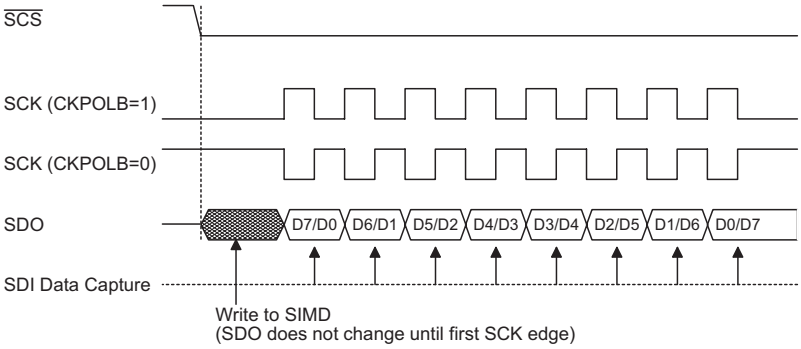
SPI Communication

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

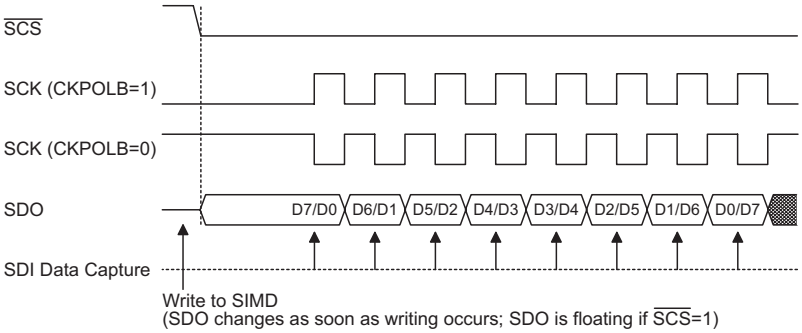
The SPI will continue to function even in the IDLE Mode.



SPI Master Mode Timing

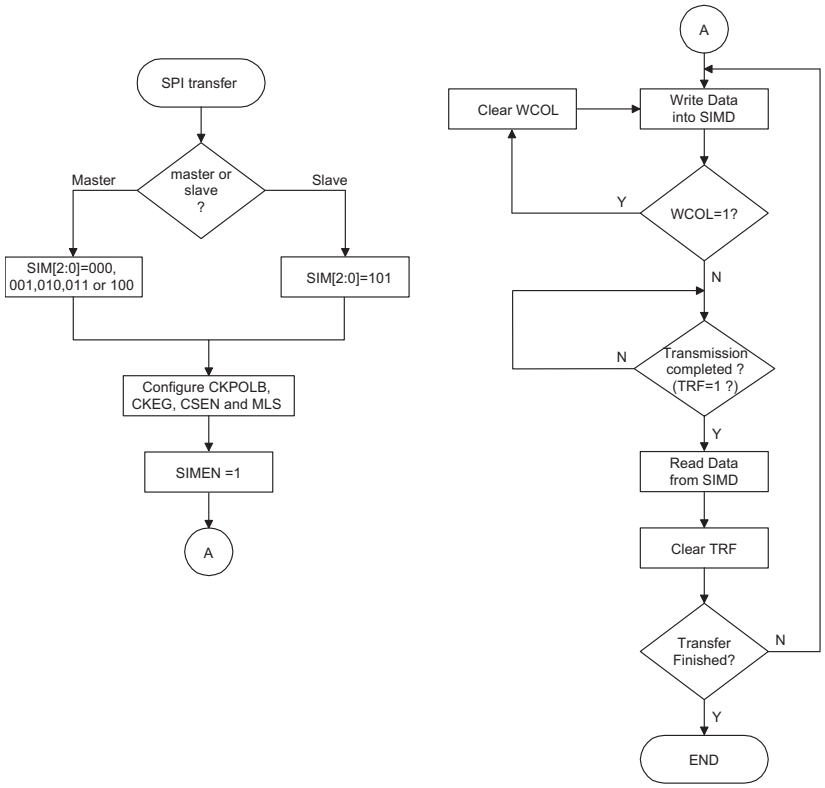


SPI Slave Mode Timing – CKEG=0



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

SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flowchart

SPI Bus Enable/Disable

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

SPI Operation

All communication is carried out using the 4-line interface for either Master or Slave Mode.The CSEN bit in the SIMC2 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{SCS}$ line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the $\overline{SCS}$ line will be an I/O pin or the other functions and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the SDI line in a floating condition and the SDO line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and $\overline{SCS}$, SDI, SDO and SCK will all become I/O pins or the other functions. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SIMD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode:

Master Mode

- Step 1
  Select the SPI Master mode and clock source using the SIM2~SIM0 bits in the SIMC0 control register.
- Step 2
  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave device.
- Step 3
  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then use the SCK and $\overline{SCS}$ lines to output the data. After this, go to step5.
  For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.
- Step 5
  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the TRF bit or wait for a SPI serial bus interrupt.
- Step 7
  Read data from the SIMD register.
- Step 8
  Clear TRF.
- Step 9
  Go to step 4.

Slave Mode

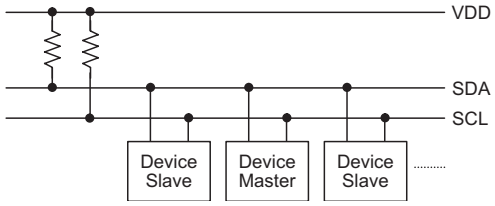
- Step 1
  Select the SPI Slave mode using the SIM2~SIM0 bits in the SIMC0 control register
- Step 2
  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master device.
- Step 3
  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and $\overline{SCS}$ signal. After this, go to step5.
  For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.
- Step 5
  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the TRF bit or wait for a SPI serial bus interrupt.
- Step 7
  Read data from the SIMD register.
- Step 8
  Clear TRF.
- Step 9
  Go to step 4.

Error Detection

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

I²C Interface

The I²C interface is used to communicate with external peripheral devices such as sensors 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.



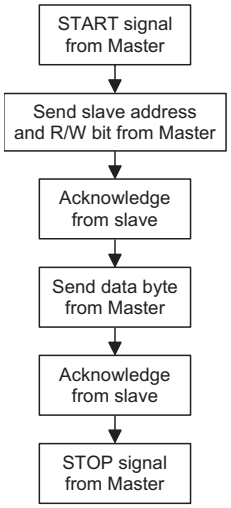
I²C Master/Slave Bus Connection

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 this device, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode.

There are several configuration options associated with the I²C interface. One of these is to enable the function which selects the SIM pins rather than normal I/O pins. Note that if the configuration option does not select the SIM function then the SIMEN bit in the SIMC0 register will have no effect. A configuration option exists to allow a clock other than the system clock to drive the I²C interface. Another configuration option determines the debounce time of the I²C interface. This uses the internal clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 1 or 2 system clocks.



I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SBSC, one address register SIMA and one data register, SIMD. The SIMD register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I²C bus. Before the microcontroller writes data to the I²C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I²C bus, the microcontroller can read it from the SIMD register. Any transmission or reception of data from the I²C bus must be made via the SIMD register.

Note that the SIMA register also has the name SIMC2 which is used by the SPI function. Bit SIMEN and bits SIM2~SIM0 in register SIMC0 are used by the I²C interface. The I2CDB0 and I2CDB1 in the SBSC register are used to select the I²C debounce time.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	D0
SBSC	SIM_WCOL	—	I2CDB1	I2CDB0	—	—	—	—

I²C Registers List

SIMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	1	1	1	0	0	0	0	—

Bit 7~5 **SIM2~SIM0:** SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{LIRC}
100: SPI master mode; SPI clock is TM0 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: I/O mode

These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from the TM0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **PCKEN:** PCK Output Pin Control
0: Disable
1: Enable

Bit 3~2 **PCKP1, PCKP0:** Select PCK output pin frequency
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/8$
11: TM0 CCRP match frequency/2

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

Bit 0 Unimplemented, read as "0"

SIMC1 Register

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

Bit 7 **HCF:** I^2C 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:** I^2C Bus address match flag
0: Not address match
1: Address match
The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB:** I^2C Bus busy flag
0: I^2C Bus is not busy
1: I^2C Bus is busy
The HBB flag is the I^2C busy flag. This flag will be "1" when the I^2C 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:** Select I^2C slave device is transmitter or receiver
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK:** I^2C Bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag
The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8 bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set TXAK bit to "0" before further data is received.

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

Bit 1 **IAMWU:** I²C Address Match Wake-up Control
0: Disable
1: Enable
This bit should be set to "1" to enable I²C address match wake up from SLEEP or IDLE Mode.

Bit 0 **RXAK:** I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave do 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.

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

SIMD Register

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

"x" unknown

SIMA Register

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	0

"x" unknown

Bit 7~1 **IICA6~IICA0:** I²C slave address

IICA6~ IICA0 is the I²C slave address bit 6 ~ bit 0.

The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~ 1 of the SIMA register define the device slave address. Bit 0 is not defined.

When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

Bit 0 Undefined bit

This bit can be read or written by user software program.

SBSC Register

Bit	7	6	5	4	3	2	1	0
Name	SIM_WCOL	—	I2CDB1	I2CDB0	—	—	—	—
R/W	R/W	—	R/W	R/W	—	—	—	—
POR	0	—	0	0	—	—	—	—

Bit 7 **SIM_WCOL:** SIM WCOL control bit

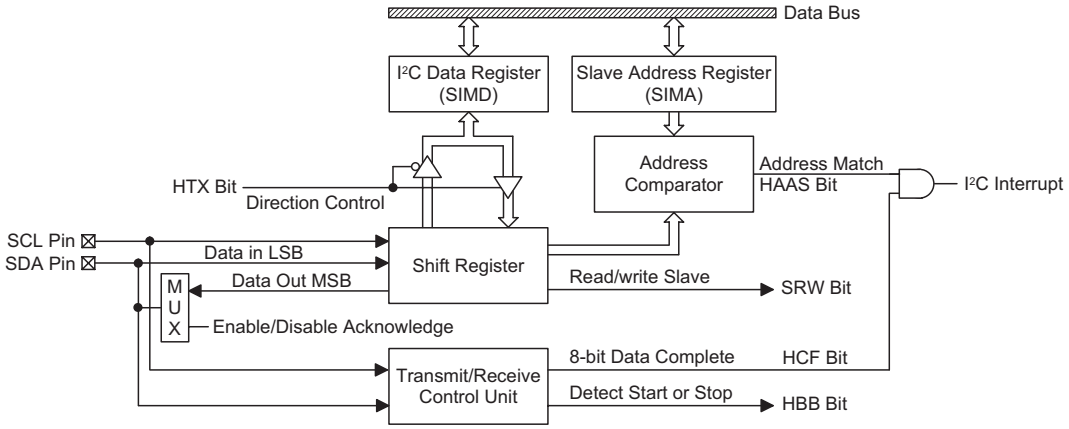
Related to SPI, described elsewhere

Bit 6 Unimplemented, read as "0"

Bit 5~4 **I2CDB1~I2CDB0:** I²C debounce selection bits

00: No debounce (default)
01: 1 system clock debounce
10: 2 system clocks debounce
11: 2 system clocks debounce

Bit 3~0 Unimplemented, read as "0"

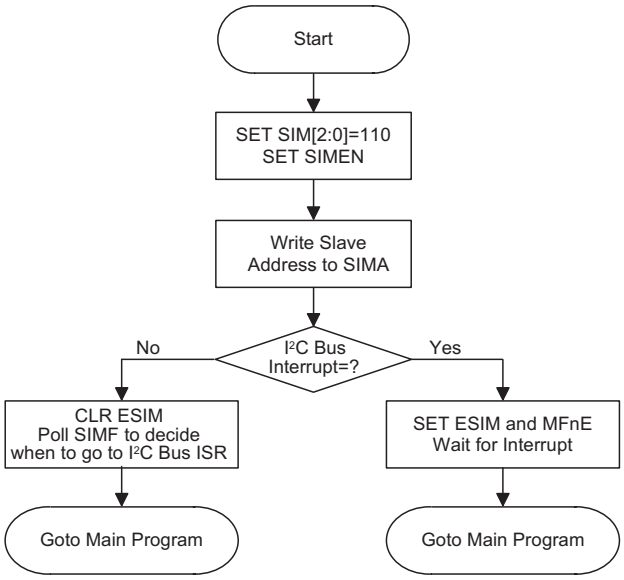


I²C Block Diagram

I²C Bus Communication

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

- Step 1
  Set the SIM2~SIM0 and SIMEN bits in the SIMC0 register to "1" to enable the I²C bus.
- Step 2
  Write the slave address of the device to the I²C bus address register SIMA.
- Step 3
  Set the SIME and SIM Muti-Function interrupt enable bit of the interrupt control register to enable the SIM interrupt and Multi-function interrupt.



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

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

I²C Bus Slave Address

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

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

I²C Bus Read/Write Signal

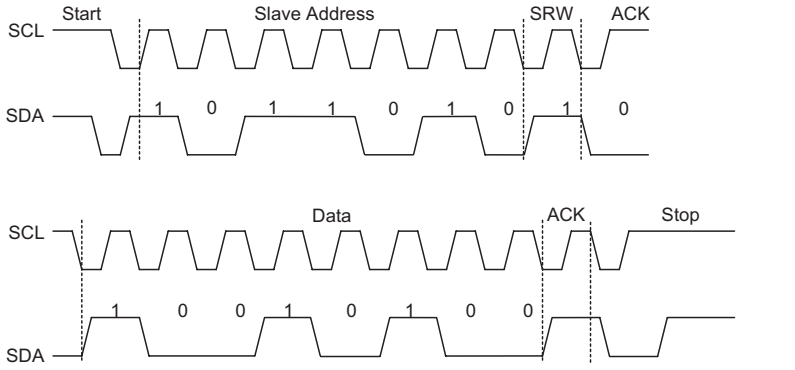
The SRW bit in the SIMC1 register defines whether the slave device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is "1" then this indicates that the master device wishes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

I²C Bus Slave Address Acknowledge Signal

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

I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register. When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the RXAK bit in the SIMC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.

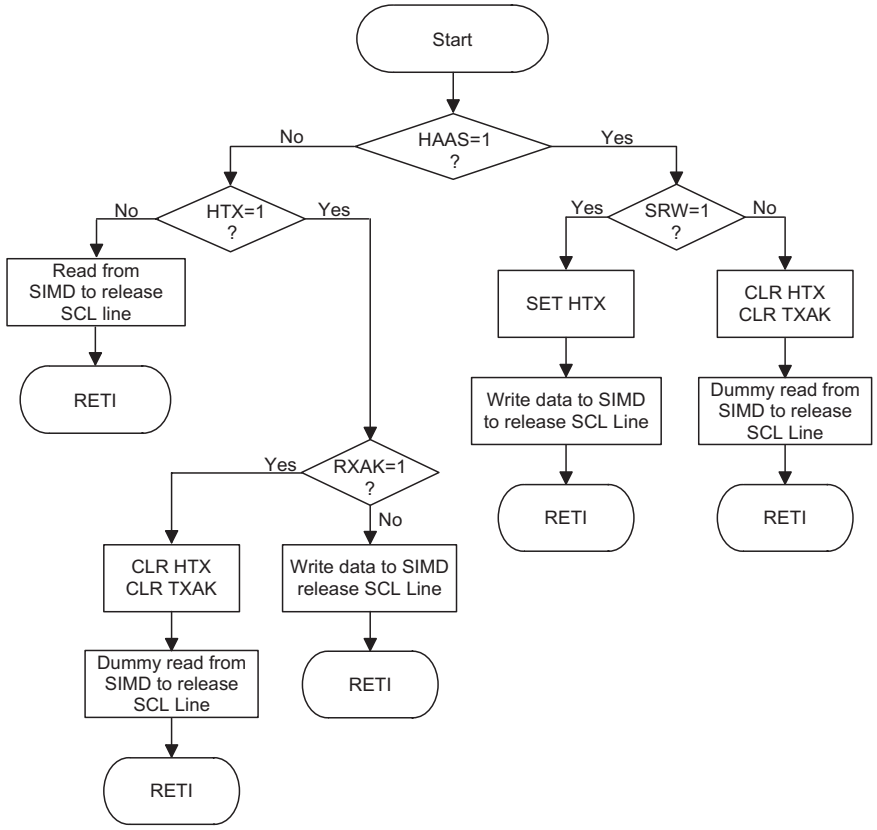


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

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

I²C Communication Timing Diagram

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



I²C Bus ISR Flow Chart

I²C Time Out Operation

The time-out counter will start counting when the I²C interface received the START bit and address match. After that the counter will be cleared on each falling edge of the SCL pin. If the time counter is larger than the selected time-out time, then the anti-locked protection scheme will take place and the time-out counter will be stopped by hardware automatically, the I2CTOF bit will be set high and an I²C interrupt will also take place. Note that this scheme can also be stopped when the I²C received the STOP bit. There are several time-out periods can be selected by the I2CTOS0~I2CTOS5 bits in the I2CTOC register.

I2CTOC Register

Bit	7	6	5	4	3	2	1	0
Name	I2CTOEN	I2CTOF	I2CTOS5	I2CTOS4	I2CTOS3	I2CTOS2	I2CTOS1	I2CTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **I2CTOEN:** I²C Time Out function control bit
0: Disable
1: Enable

Bit 6 **I2CTOF:** I²C Time Out indication bit
0: Not occurred
1: Occurred

Bit 5~0 **I2CTOS5~I2CTOS0:** I²C Time out time period select
The I²C Time out clock is provided by the $f_L/32$. The time out time period can be calculated from the accompanying equation.
([I2CTOS5:I2CTOS0] + 1) × ($32/f_L$)

Peripheral Clock Output

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

Peripheral Clock Operation

As the peripheral clock output pin, PCK, is shared with I/O line, the required pin function is chosen via PCKEN in the SIMC0 register. The Peripheral Clock function is controlled using the SIMC0 register. The clock source for the Peripheral Clock Output can originate from either the TM0 CCRP match frequency/2 or a divided ratio of the internal f_{SYS} clock. The PCKEN bit in the SIMC0 register is the overall on/off control, setting PCKEN bit to "1" enables the Peripheral Clock, setting PCKEN bit to "0" disables it. The required division ratio of the system clock is selected using the PCKP1 and PCKP0 bits in the same register. If the device enters the SLEEP Mode, this will disable the Peripheral Clock output.

SIMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	1	1	1	0	0	0	0	—

Bit 7~5 **SIM2~SIM0:** SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{LIRC}
100: SPI master mode; SPI clock is TM0 CCRP match frequency/2
101: SPI slave mode
110: I^2C slave mode
111: I/O mode
These bits setup the overall operating mode of the SIM function. As well as selecting if the I^2C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from the TM0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **PCKEN:** PCK Output Pin Control
0: Disable
1: Enable

Bit 3~2 **PCKP1, PCKP0:** Select PCK output pin frequency
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/8$
11: TM0 CCRP match frequency/2

Bit 1 **SIMEN:** SIM Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will be in a floating condition and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. Note that when the SIMEN bit changes from low to high the contents of the SPI control registers will be in an unknown condition and should therefore be first initialised by the application program.

Bit 0 Unimplemented, read as "0"

Pulse Width Modulator

The device contains three Pulse Width Modulation outputs, PWM0, PWM1 and PWM2. Useful for such applications such as motor speed control, the PWM function provides outputs with a fixed frequency but with a duty cycle that can be varied by setting particular values into the corresponding PWM register.

PWM Operation

The device has three PWM registers, PWM0, PWM1 and PWM2, where the 8-bit value, which represents the overall duty cycle of one modulation cycle of the output waveform, should be placed. To increase the PWM modulation frequency, each modulation cycle is modulated into four/two individual modulation sub-sections, known as the 6+2/7+1 mode. Note that it is only necessary to write the required modulation value into the corresponding PWM register as the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware. For the device, the PWM clock source is the system clock f_{SYS}.

This method of dividing the original modulation cycle into a further 2/4 sub-cycles enables the generation of higher PWM frequencies, which allow a wider range of applications to be served. As long as the periods of the generated PWM pulses are less than the time constants of the load, the PWM output will be suitable as such long time constant loads will average out the pulses of the PWM output.

PWM Modulation Frequency	PWM Cycle Frequency	PWM Cycle Duty
f_{SYS}/64 for (6+2) bits mode f_{SYS}/128 for (7+1) bits mode	f_{SYS}/256	[PWM]/256

PWMC Register

Bit	7	6	5	4	3	2	1	0
Name	PWMM	—	—	—	—	PWM2C	PWM1C	PWM0C
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **PWMM:** PWM mode selection
0: (6+2) mode
1: (7+1) mode

Bit 6~3 Unimplemented, read as "0"

Bit 2 **PWM2C:** PWM2 control bit
0: Disable
1: Enable

Bit 1 **PWM1C:** PWM1 control bit
0: Disable
1: Enable

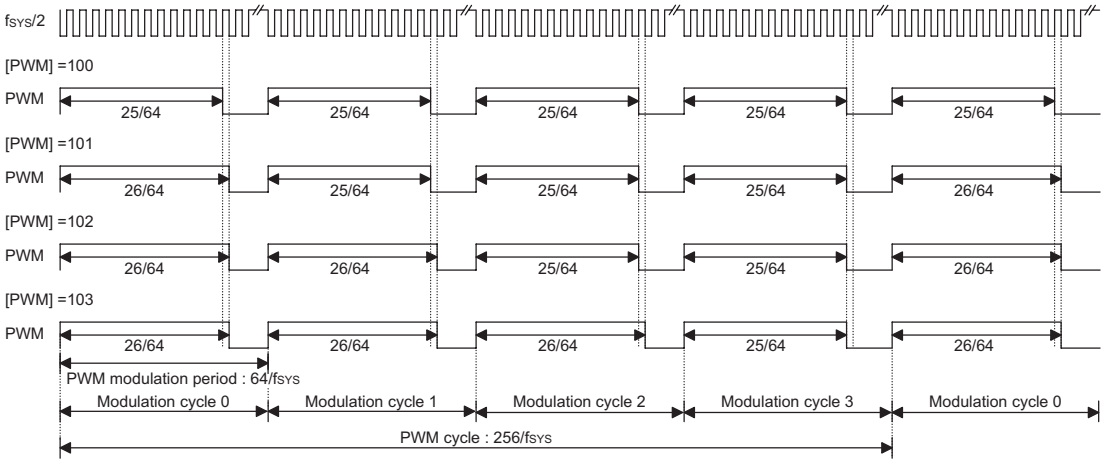
Bit 0 **PWM0C:** PWM0 control bit
0: Disable
1: Enable

6+2 PWM Mode

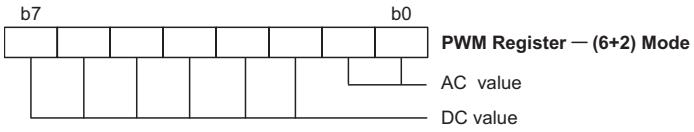
Each full PWM cycle, as it is controlled by an 8-bit PWM register, has 256 clock periods. However, in the 6+2 PWM Mode, each PWM cycle is subdivided into four individual sub-cycles known as modulation cycle 0 ~ modulation cycle 3, denoted as "i" in the table. Each one of these four sub-cycles contains 64 clock cycles. In this mode, a modulation frequency increase by a factor of four is achieved. The 8-bit PWM, PWM0, PWM1 or PWM2 register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit 2~bit 7 is denoted here as the DC value. The second group which consists of bit 0~bit 1 is known as the AC value. In the 6+2 PWM mode, the duty cycle value of each of the four modulation sub-cycles is shown in the following table.

The following diagram illustrates the waveforms associated with the 6+2 mode of PWM operation. It is important to note how the single PWM cycle is subdivided into 4 individual modulation cycles, numbered from 0~3 and how the AC value is related to the PWM value.

Parameter	AC (0~3)	DC (Duty Cycle)
Modulation cycle i (i=0~3)	i < AC	(DC+1)/64
	i ≥ AC	DC/64



6+2 PWM Mode

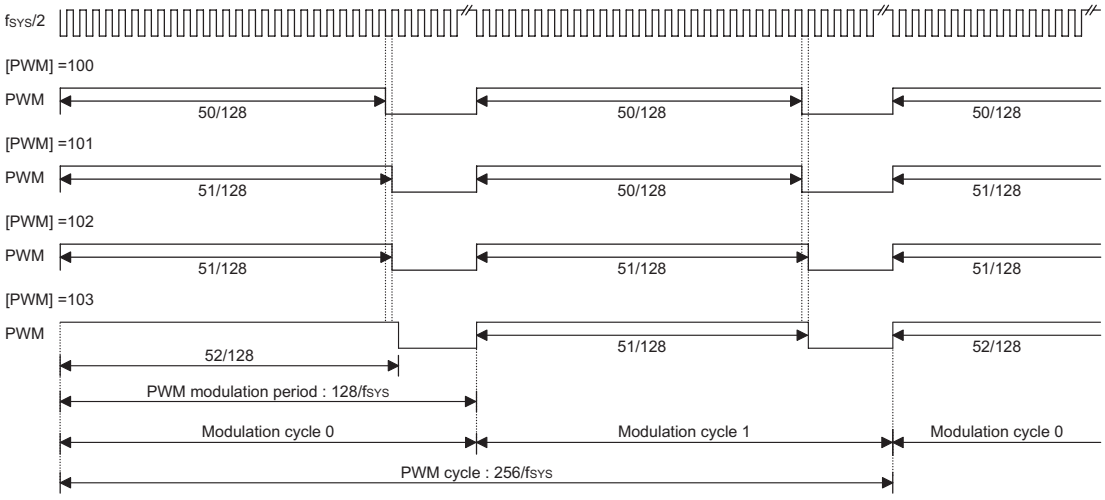


PWM Register for 6+2 Mode

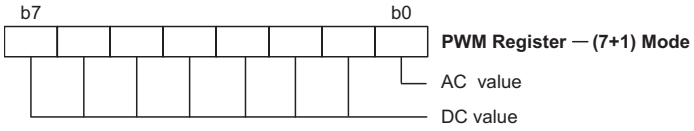
7+1 PWM Mode

Each full PWM cycle, as it is controlled by an 8-bit PWM register, has 256 clock periods. However, in the 7+1 PWM mode, each PWM cycle is subdivided into two individual sub-cycles known as modulation cycle 0 ~ modulation cycle 1, denoted as "i" in the table. Each one of these two sub-cycles contains 128 clock cycles. In this mode, a modulation frequency increase of two is achieved. The 8-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit 1~bit 7 is denoted here as the DC value. The second group which consists of bit 0 is known as the AC value. In the 7+1 PWM mode, the duty cycle value of each of the two modulation sub-cycles is shown in the following table.

Parameter	AC (0~1)	DC (Duty Cycle)
Modulation cycle i (i=0~1)	i < AC	(DC+1)/128
	i ≥ AC	DC/128



7+1 PWM Mode



PWM Register for 7+1 Mode

PWM Output Control

On the device, the PWM outputs are pin-shared with pins PC4, PC5 and PC6. To operate as PWM outputs and not as I/O pins, the correct PWM configuration options must be selected. A "0" must also be written to the corresponding bits in the I/O port control register PCC to ensure that the required PWM output pin is setup as an output. After these two initial steps have been carried out, and of course after the required PWM value has been written into the PWM register, writing a "1" to the corresponding bit in the PC output data register will enable the PWM data to appear on the pin. Writing a "0" to the corresponding bit in the PC output data register will disable The PWM output function and force the output low. In this way, the Port C data output register can be used as an on/off control for the PWM function. Note that if the configuration options have selected the PWM function, but a "1" has been written to its corresponding bit in the PCC control register to configure the pin as an input, then the pin can still function as a normal input line, with pull-high resistor options.

PWM Programming Example

The following sample program shows how the PWM outputs are setup and controlled. Before use the corresponding PWM output configuration options must first be selected.

```
mov a,64h          ; setup PWM value of 100 decimal which is 64H
mov pwm0,a
set pwmc.7         ; select the 7+1 PWM mode
set pwmc.0         ; setup PWM0 shared with PC4
clr pcc.4          ; setup pin PC4 as an output
set pc.4           ; PC.4=1; enable the PWM output
:
:
clr pc.4           ; disable the PWM output _ PC4 will remain low
```

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions. The external interrupt is generated by the action of the external INTn pin, while the internal interrupts are generated by various internal functions such as TMs, LVD, SIM and USB.

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 first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0 register 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 registers as well as interrupt flags to indicate the presence of an interrupt request. The naming convention of these follows a specific pattern. First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an "E" for enable/disable bit or "F" for request flag.

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pin	INTnE	INTnF	n=0~1
Multi-function	MF0E	MF0F	—
SIM	SIME	SIMF	—
LVD	LVE	LVF	—
TM	TnPE	TnPF	n=0~1
	TnAE	TnAF	n=0~1
USB SIE	USBE	USBF	—
USB Setup Token	USTE	USTF	—
USB Endpoint 0 IN Token	UITE	UITF	—
USB Endpoint 0 OUT Token	UOTE	UOTF	—

Interrupt Register Bit Naming Conventions

Interrupt Register Contents

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	USBF	INT1F	INT0F	USBE	INT1E	INT0E	EMI
INTC1	—	UOTF	UITF	USTF	—	UOTE	UITE	USTE
INTC2	—	LVF	SIMF	MF0F	—	LVE	SIME	MF0E
MFI0	T1AF	T1PF	T0AF	T0PF	T1AE	T1PE	T0AE	T0PE

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: both rising and falling edges

Bit 1~0 **INT0S1~INT0S0:** interrupt edge control for INT0 pin
00: disable
01: rising edge
10: falling edge
11: both rising and falling edges

INTC0 Register

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

Bit 7 Unimplemented, read as "0"

Bit 6 **USBF:** USB SIE Interrupt Request Flag
0: No request
1: Interrupt request

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

Bit 4 **INT0F:** INT0 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **USBE:** USB SIE Interrupt Control
0: Disable
1: Enable

Bit 2 **INT1E:** INT1 Interrupt Control
0: Disable
1: Enable

Bit 1 **INT0E:** INT0 Interrupt Control
0: Disable
1: Enable

Bit 0 **EMI:** Global Interrupt Control
0: Disable
1: Enable

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	UOTF	UITF	USTF	—	UOTE	UITE	USTE
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 **UOTF:** USB Endpoint 0 OUT Token Interrupt Request Flag
0: No request
1: Interrupt request
Bit 5 **UITF:** USB Endpoint 0 IN Token Interrupt Request Flag
0: No request
1: Interrupt request
Bit 4 **USTF:** USB Setup Token Interrupt Request Flag
0: No request
1: Interrupt request
Bit 3 Unimplemented, read as "0"
Bit 2 **UOTE:** USB Endpoint 0 OUT Token Interrupt Control
0: Disable
1: Enable
Bit 1 **UITE:** USB Endpoint 0 IN Token Interrupt Control
0: Disable
1: Enable
Bit 0 **USTE:** USB Setup Token Interrupt Control
0: Disable
1: Enable

INTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	LVF	SIMF	MF0F	—	LVE	SIME	MF0E
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 **LVF:** LVD Interrupt Request Flag
0: No request
1: Interrupt request
Bit 5 **SIMF:** SIM Interrupt Request Flag
0: No request
1: Interrupt request
Bit 4 **MF0F:** Multi-function Interrupt 0 Request Flag
0: No request
1: Interrupt request
Bit 3 Unimplemented, read as "0"
Bit 2 **LVE:** LVD Interrupt Control
0: Disable
1: Enable
Bit 1 **SIME:** SIM Interrupt Control
0: Disable
1: Enable
Bit 0 **MF0E:** Multi-function Interrupt 0 Control
0: Disable
1: Enable

MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	T1AF	T1PF	T0AF	T0PF	T1AE	T1PE	T0AE	T0PE
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 **T1AF:** TM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **T1PF:** TM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 5 **T0AF:** TM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T0PF:** TM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 **T1AE:** TM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **T1PE:** TM1 Comparator P match interrupt control
0: Disable
1: Enable

Bit 1 **T0AE:** TM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **T0PE:** TM0 Comparator P match interrupt control
0: Disable
1: Enable

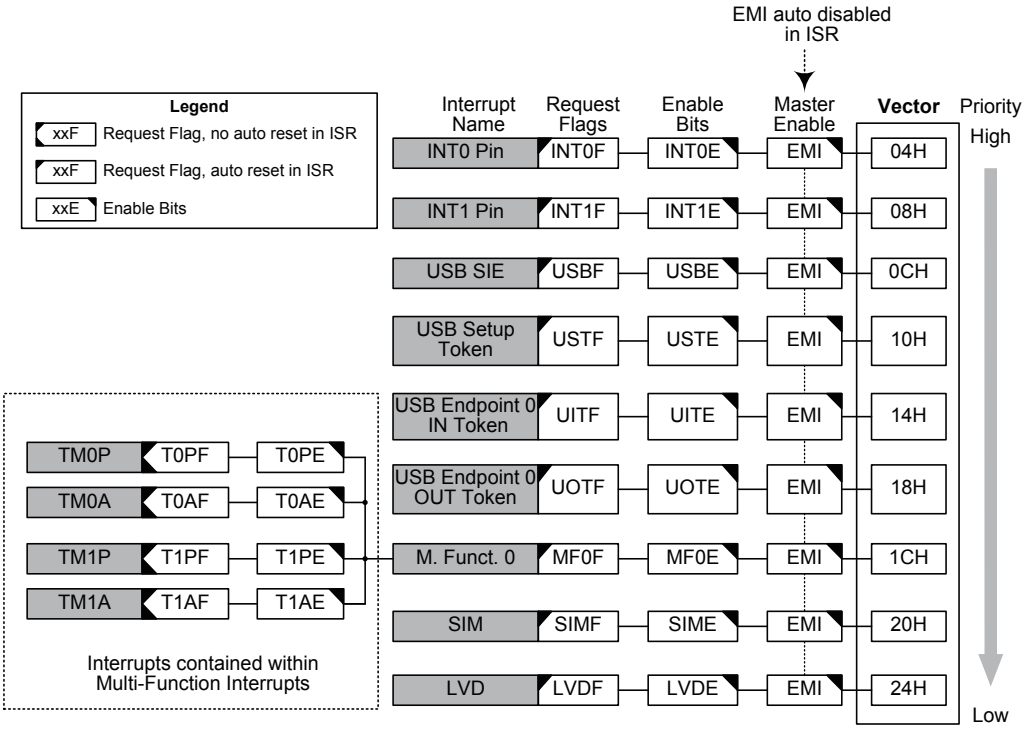
Interrupt Operation

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

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

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

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



Interrupt Structure

External Interrupt

The external interrupt is controlled by signal transitions on the INTn pins. An external interrupt request will take place when the external interrupt request flag, INTnF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INTnE, must first be set. Additionally the correct interrupt edge type must be selected using the related register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if the external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTnF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pin will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

USB SIE Interrupt

A USB SIE interrupt request will take place when the USB SIE interrupt request flags, USBF, is set, a situation that will occur when an endpoint except endpoint 0 is accessed. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and USB SIE 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 SIE interrupt vector, will take place. When the interrupt is serviced, the USB SIE interrupt request flag, USBF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

USB Setup Token Interrupt

A USB Setup Token interrupt request will take place when the USB Setup Token interrupt request flags, USTF, is set, a situation that will occur when the USB receives endpoint 0 Setup token signal from PC. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and USB Setup Token interrupt enable bit, USTE, 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 Setup Token interrupt vector, will take place. When the interrupt is serviced, the USB Setup Token interrupt request flag, USTF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

USB Endpoint 0 IN Token Interrupt

A USB Endpoint 0 IN Token interrupt request will take place when the USB Endpoint 0 IN Token interrupt request flags, UITF, is set, a situation that will occur when the USB receives endpoint 0 IN token signal from PC. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and USB Endpoint 0 IN Token interrupt enable bit, UITE, 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 Endpoint 0 IN Token interrupt vector, will take place. When the interrupt is serviced, the USB Endpoint 0 IN Token interrupt request flag, UITF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

USB Endpoint 0 OUT Token Interrupt

A USB Endpoint 0 OUT Token interrupt request will take place when the USB Endpoint 0 OUT Token interrupt request flags, UOTF, is set, a situation that will occur when the USB receives endpoint 0 OUT token signal from PC. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and USB Endpoint 0 OUT Token interrupt enable bit, UOTE, 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 Endpoint 0 OUT Token interrupt vector, will take place. When the interrupt is serviced, the USB Endpoint 0 OUT Token interrupt request flag, UOTF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Serial Interface Module Interrupt

The Serial Interface Module interrupt, known as the SIM interrupt, will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, SIME, must first be set. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SIM interface, a subroutine call to the respective Interrupt vector, will take place. When the Serial Interface Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the SIMF flag will be automatically cleared as well.

LVD Interrupt

The Low Voltage Detector interrupt, known as the LVD interrupt, 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 and Low Voltage Interrupt enable bit, LVE, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the LVD Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the LVF flag will be automatically cleared as well.

Multi-function Interrupt

Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupts.

A Multi-function interrupt request will take place the Multi-function interrupt request flag, MF0F is set. The Multi-function interrupt flag will be set when any of its included functions generate an interrupt request flag. 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 the Multi-function interrupt vector will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

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

TM Interrupts

The Compact Type TMs have two interrupts each. All of the TM interrupts are contained within the Multi-function Interrupts. For each of the Compact Type TMs there are two interrupt request flags TnPF and TnAF and two enable bits TnPE and TnAE. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

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

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins, 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 SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

Low Voltage Detector – LVD

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

LVD Register

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

LVDC Register

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

Bit 7~6 Unimplemented, read as "0"

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

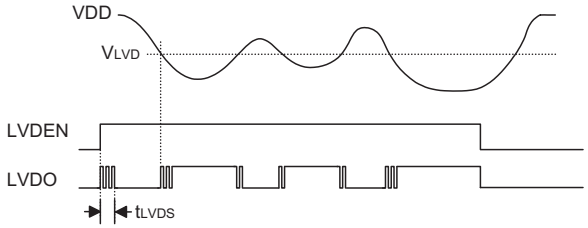
Bit 4 **LVDEN:** Low Voltage Detector Control
0: Disable
1: Enable

Bit 3 Unimplemented, read as "0"

Bit 2~0 **VLVD2~VLVD0:** Select LVD Voltage
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.

When LVD function is enabled, it is recommenced to clear LVD flag first, and then enables interrupt function to avoid mistake action.

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 device includes a USB interface function allowing for the convenient design of USB peripheral products.

Power Plane

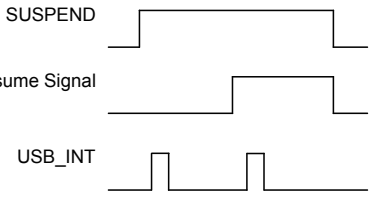
There is one power plane for HT68FB240: UBUS/VDD.

For the UBUS/VDD will supply all the HT68FB240 circuits including USB SIE and be sourced from pin "UBUS/VDD". Once the USB is removed from the USB and there is no power in the UBUS/VDD pin, the HT68FB240 circuit is no longer operational.

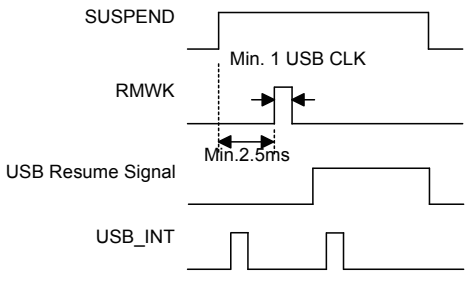
USB Suspend Wake-Up Remote Wake-Up

If there is no signal on the USB bus for over 3ms, the device will go into a 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 spec. 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 current can be further decreased by setting the SUSP2 bit in the UCC register. When the resume signal is sent out by the host, the device will be woken up the by the USB interrupt and the RESUME bit in the USC register will be set. To ensure correct device operation, the program must set the USBCKEN bit in the UCC register high and clear the SUSP2 bit in the UCC register. The Resume signal will be cleared before the Idle signal is sent out by the host and the Suspend line in the USC register will change to zero. So when the MCU detects the Suspend bit in the USC register, the condition of the Resume line should be noted and taken into consideration.



The device has a remote wake up function which can wake-up the USB Host by sending a wake-up pulse through RMWK in the USC register. Once the USB Host receives a wake-up signal from the device, it will send a Resume signal to the device.



USB Interface Operation

The device has 3 Endpoints (EP0~EP2). EP0 supports Control transfer for Setup, IN and OUT token respectively. EP1 and EP2 support Interrupt transfer. These registers, including USB_STAT, UINT, USC, UCC, AWR, STL, SIES, MISC, UFEN, URDCT and FIFO0~FIFO2 are used for the USB function.

EP0, EP1 and EP2 each have 8-byte FIFO size respectively. EP0 supports DMA interface for Configure USB descriptor. Please reference the DMA for detail description. EP0~EP2 are equiped with independent registers and not shared with the general data memory RAM. The Serial Interface Engine (SIE) of the device is compatible with USB 2.0 Low speed protocol.

The URD in the USC register is the USB reset signal control function definition bit.

USB Interface Registers

The USB interface has a series of registers associated with its operation.

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	x	x	0	0	0	—

"x" unknown

"-" Hardware did not give ESD bit initial value in power on reset

Bit 7 **PS2_CKO:** Output for driving UDP/GPIO1 pin, when work under 3D PS2 mouse function. Default value is "1".

Bit 6 **PS2_DAO:** Output for driving UDN/GPIO0 pin, when work under 3D PS2 mouse function. Default value is "1".

Bit 5 **PS2_CKI:** UDP/GPIO1 input.

Bit 4 **PS2_DAI:** UDN/GPIO0 input.

Bit 3 **SE1:** This bit is used to indicate the SIE has detected a SE1 noise in the USB bus. This bit is set by SIE and cleared by F/W.

Bit 2 **SE0:** This bit is used to indicate the SIE has detected a SE0 noise in the USB bus. This bit is set by SIE and cleared by F/W.

Bit 1 **PU:** Bit1=1, UDP and UDN have a 600kΩ pull-high. Bit1=0, no pull-high (default on MCU reset)

Bit 0 **ESD:** This bit will set to "1" when there is ESD issue.
This bit is set by SIE and cleared by F/W. Hardware did not give ESD bit initial value in power on reset.

The endpoint requestflags (EP0F, EP1F and EP2F) 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.

UINT Register

Bit	7	6	5	4	3	2	1	0
Name	—	EP2F	EP1F	EP0F	—	EP2EN	EP1EN	EP0EN
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 **EP2F:** USB endpoint2 accessed detection
0: no accessed
1: accessed

Bit 5 **EP1F:** USB endpoint1 accessed detection
0: no accessed
1: accessed

Bit 4 **EP0F:** USB endpoint0 accessed detection
0: no accessed
1: accessed

Bit 3 Unimplemented, read as "0"

Bit 2 **EP2EN:** USB endpoint2 interrupt control bit
0: disable
1: enable

Bit 1 **EP1EN:** USB endpoint1 interrupt control bit
0: disable
1: enable

Bit 0 **EP0EN:** USB endpoint0 interrupt control bit
0: disable
1: enable

USC Register

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

"x" unknown

Bit 7 **URD:** USB reset signal control function definition
0: USB reset signal cannot MCU
1: USB reset signal will reset MCU

Bit 6 **SELPS2:** PS2 mode select bit
0: not PS2 mode
1: PS2 mode
When the SELPS2 bit is set high, the PS2 function is selected and the pin-shared pins, UDN/GPIO0 and UDP/GPIO1, will become the GPIO0 and GPIO1 general purpose I/O functions which can be used to be the DATA and CLK pins for the PS2.

Bit 5 Unimplemented, read as "0"

Bit 4 **SELUSB:** USB mode and V33O on/off select bit
0: not USB mode, turn-off V33O
1: USB mode, turn-on V33O
When the V33C bit is set high, the USB and V33O functions is selected and the pin-shared pins, UDN/GPIO0 and UDP/GPIO1, will become the UDN and UDP pins for the USB.

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.V33O 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

x: don't care

Bit 3 **RESUME:** USB resume indication bit
0: SUSP bit goes to "0"
1: leave the suspend mode
When the USB leaves the suspend mode, this bit is set to "1" (set by SIE). When the RESUME is set by SIE, an interrupt will be generated to wake-up the MCU. In order to detect the suspend state, the MCU should set USBCKEN and clear SUSP2 (in the UCC register) to enable the SIE detect function. RESUME will be cleared when the SUSP goes to "0". When the MCU is detecting the SUSP, the condition of RESUME (causes the MCU to wake-up) should be noted and taken into consideration.

Bit 2 **URST:** USB reset indication bit
0: no USB reset
1: USB reset occurred
This bit is set/cleared by the USB SIE. This bit is used to detect a USB reset event on the USB bus. When this bit is set to "1", this indicates that a USB reset has occurred and that a USB interrupt will be initialized.

Bit 1 **RMWK:** USB remote wake-up command
0: no remote wake-up
1: remote wake-up
It is set by MCU to leave the USB host leaving the suspend mode. Indicate that the USB host leaves the suspend mode.

Bit 0 **SUSP:** USB suspend indication
0: not in the suspend mode
1: enter the suspend mode
When this bit is set to 1 (set by SIE), it indicates that the USB bus has entered the suspend mode. The USB interrupt is also triggered when this bit changes from low to high.

UCC Register

Bit	7	6	5	4	3	2	1	0
Name	Rctrl	SYSCLK	—	SUSP2	USBCKEN	—	EPS1	EPS0
R/W	R/W	R/W	—	R/W	R/W	—	R/W	R/W
POR	0	0	—	0	0	—	x	x

"x" unknown

Bit 7 **Rctrl:** 7.5kΩ resistor between UDN and UBUS control bit
0: no 7.5kΩ resistor between UDN and UBUS
1: has 7.5kΩ resistor between UDN and UBUS

Bit 6 **SYSCLK:** Specify MCU oscillator frequency indication bit
0: 12MHz crystal oscillator or resonator
1: 6MHz crystal oscillator or resonator

Bit 5 Unimplemented, read as "0"

Bit 4 **SUSP2:** Reduce power consumption in suspend mode control bit
0: in normal mode
1: in halt mode, set this bit to "1" for reducing power consumption

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

Bit 2 Unimplemented, read as "0"

Bit 1~0 **EPS1, EPS0:** Accessing endpoint FIFO selection
00: select endpoint 0 FIFO (control)
01: select endpoint 1 FIFO
10~11: select endpoint 2 FIFO

AWR 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	x	x	x	x	x	x	x	x

"x" unknown

Bit 7~1 **AD6~AD0:** USB device address

Bit 0 **WKEN:** USB remote-wake-up control bit
0: Disable
1: Enable

The AWR register contains the current address and a remote wake up function control bit. The initial value of AWR is "00H". The address value extracted from the USB command has not to be loaded into this register until the SETUP stage has finished

STL Register

Bit	7	6	5	4	3	2	1	0
Name	—	STLO2	STLO1	STLO0	—	STLI2	STLI1	STLI0
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	x	x	x	—	x	x	x

"x" unknown

Bit 7 Unimplemented, read as "0"

Bit 6~4 **STLO3~STLO0:** FIFO OUT stall endpoints indication bits
0: not stall
1: stall
The STL register shows if the corresponding endpoint has worked properly or not. As soon as endpoint improper operation occurs, the related bit in the STL register has to be set high. The STL register bits will be cleared by a USB reset signal and a setup token event.

Bit 3 Unimplemented, read as "0"

Bit 2~0 **STLI3~STLI0:** FIFO IN stall endpoints indication bits
0: not stall
1: stall
The STL register shows if the corresponding endpoint has worked properly or not. As soon as endpoint improper operation occurs, the related bit in the STL register has to be set high. The STL register bits will be cleared by a USB reset signal and a setup token event.

SIES Register

Bit	7	6	5	4	3	2	1	0
Name	NMI	CRCF	—	NAK	IN	OUT	ERR	ASET
R/W	R/W	R/W	—	R	R	R/W	R/W	R/W
POR	x	x	—	x	x	x	x	x

"x" unknown

Bit 7 **NMI:** NAK token interrupt mask flag
0: interrupt enable
1: interrupt disable
If this bit set, when the device sent a NAK token to the host, an interrupt will be disabled. Otherwise if this bit is cleared, when the device sends a NAK token to the host, it will enter the interrupt sub-routine. This bit is used for all endpoint.

Bit 6 **CRCF:** CRC error detection flag
0: no error
1: error
This bit will be set to "1" when there are the following three conditions happened:
CRC error, PID error, Bit stuffing error. This bit is set by SIE and cleared by F/W.

Bit 5 Unimplemented, read as "0"

Bit 4 **NAK:** ACK error detection flag
0: no error
1: error
This bit will set to "1" once SIE discover there are some error condition so the SIE is not response (NAK or ACK or DATA) for the USB token. This bit is set by SIE and cleared by F/W.

Bit 3 **IN:** Current USB receiving signal indicator
0: low
1: high
This bit is used to indicate the current USB receiving signal from PC host is IN token.

Bit 2 **OUT:** USB OUT token indicator
0: low
1: high
This bit is used to indicate the OUT token (except the OUT zero length token) has been received. The firmware clears this bit after the OUT data has been read. Also, this bit will be cleared by SIE after the next valid SETUP token is received.

Bit 1 **ERR:** FIFO accessed error indicator
0: no error
1: error
This bit is used to indicate that some errors have occurred when the FIFO is accessed. This bit is set by SIE and should be cleared by firmware. This bit is used for all endpoint.

Bit 0 **ASET:** device address updated method control bit
0: update address after an written address to the AWR register
1: update address after PC host read out data
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 PC host has successfully read the data from he 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. So, in order to work 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	—	—	CLEAR	TX	REQUEST
R/W	R	R	R/W	—	—	R/W	R/W	R/W
POR	x	x	x	—	—	x	x	x

"x" unknown

Bit 7 **LEN0:** 0-sized packet indication flag
0: not 0-sized packet
1: 0-sized packet
This bit is used to show that the host sent a 0-sized packet to the MCU. This bit must be cleared by a read action to the corresponding FIFO.

Bit 6 **READY:** Desired FIFO ready indication flag
0: not ready
1: ready

Bit 5 **SETCMD:** Setup command indication flag
0: not setup command
1: setup command
This bit is used to show that the data in the FIFO is a setup command. This bit is set by Hardware and cleared by Firmware.

Bit 4~3 Unimplemented, read as "0"

Bit 2 **CLEAR:** Clear FIFO function control bit
0: disable
1: enable
MCU requests to clear the FIFO, even if the FIFO is not ready. After clearing the FIFO, the USB interface will send force_tx_err to tell the Host that data under-run if the Host wants to read data.

Bit 1 **TX:** data writing to FIFO status indication flag
0: data writing finished
1: data writing to FIFO
To represent the direction and transition end MCU access. When set to logic 1, the MCU desires to write data to the FIFO. After finishing, this bit must be set to logic 0 before terminating request to represent transition end. For an MCU read operation, this bit must be set to logic 0 and set to logic 1 after finishing.

Bit 0 **REQUEST:** Desired FIFO request status indication flag
0: no request
1: request
After setting the status of the desired one, FIFO can be requested by setting this bit high.After finishing, this bit must be set low.

UFEN Register

Bit	7	6	5	4	3	2	1	0
Name	—	SETO2	SETO1	—	—	SETI2	SETI1	DATATG
R/W	—	R/W	R/W	—	—	R/W	R/W	R/W
POR	—	0	0	—	—	0	0	0

Bit 7 Unimplemented, read as "0"
Bit 6 **SETO2:** EP2 output FIFO control bit
0: disable
1: enable
Bit 5 **SETO1:** EP1 output FIFO control bit
0: disable
1: enable
Bit 4~3 Unimplemented, read as "0"
Bit 2 **SETI2:** EP2 input FIFO control bit
0: disable
1: enable
Bit 1 **SETI1:** EP1 input FIFO control bit
0: disable
1: enable
Bit 0 **DATATG:** DATA token toggle bit
0: low
1: high
Note: This register will be reset only by power-on reset.

USB endpoint accessing registers

Register Name	Bit							
	7	6	5	4	3	2	1	0
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

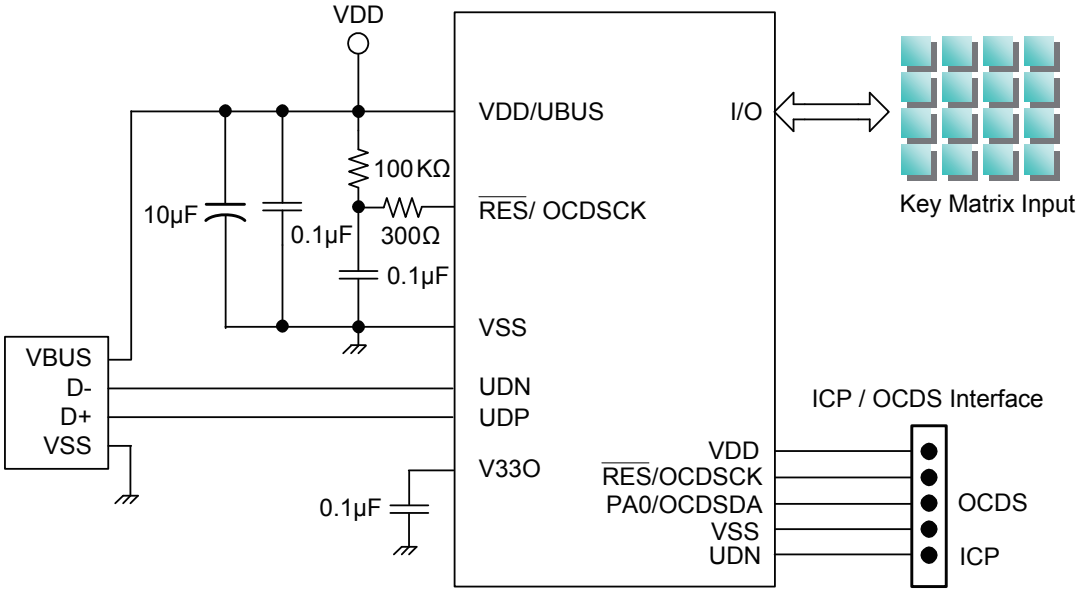
URDCT Register

Bit	7	6	5	4	3	2	1	0
Name	URDCT7	URDCT6	URDCT5	URDCT4	URDCT3	URDCT2	URDCT1	URDCT0
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 **URDCT7~URDCT0:** Used for counting the received data size of USB Setup or OUT token packets

Application Circuits



HT68FB240

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 setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

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

Instruction Set Summary

The following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

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

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch			
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].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			
TABRD [m]	Read table to TBLH and Data Memory	2Note	None
TABRDL [m]	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
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

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

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

3. For the "CLR WDT1" and "CLR WDT2" instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both "CLR WDT1" and "CLR WDT2" instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

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

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

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

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

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

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

CLR WDT1 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

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 $ACC \leftarrow \overline{[m]}$

Affected flag(s) Z

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

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

Operation [m] ← 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

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

SDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

SNZ [m].i Skip if bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

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

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

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

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 (current page) to TBLH and Data Memory

Description The low byte of 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

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

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

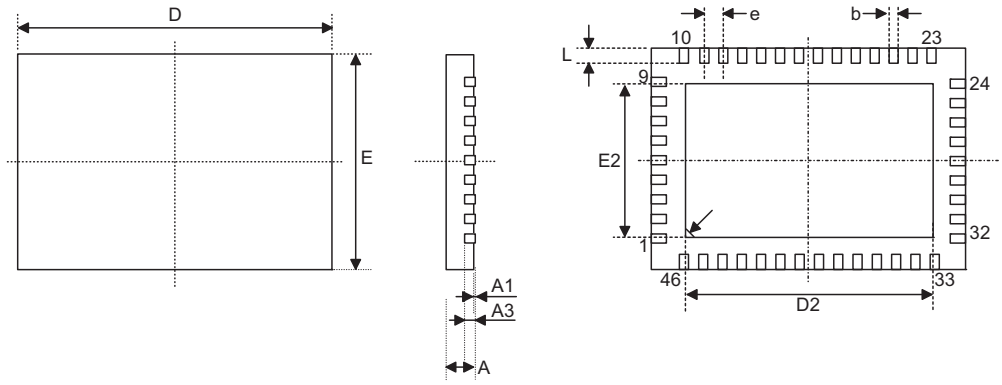
Package Information

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

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

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

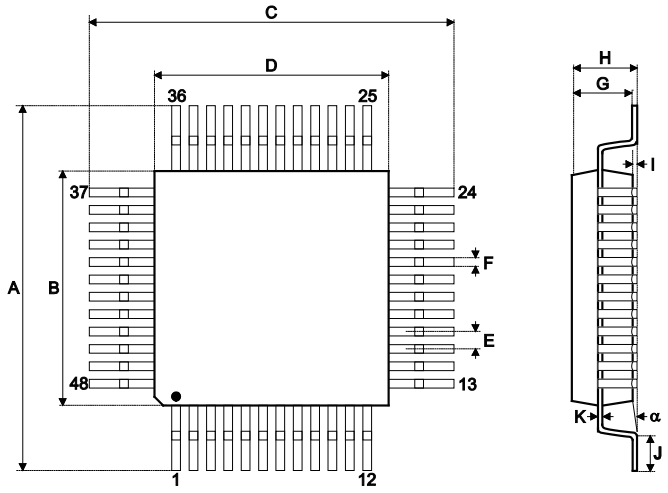
SAW Type 46-pin QFN (6.5mm×4.5mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.031	0.033	0.035
A1	0.000	0.001	0.002
A3	—	0.08 REF	—
b	0.006	0.008	0.010
D	0.254	0.256	0.258
E	0.175	0.177	0.179
e	—	0.016 BSC.	—
D2	0.197	0.201	0.205
E2	0.118	0.122	0.126
L	0.012	0.016	0.020
K	—	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.8	0.85	0.9
A1	0	0.02	0.04
A3	—	0.2 REF	—
b	0.15	0.2	0.25
D	6.45	6.5	6.55
E	4.45	4.5	4.55
e	—	0.4 BSC.	—
D2	5	5.1	5.2
E2	3	3.1	3.2
L	0.3	0.4	0.5
K	—	—	—

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°