USB Flash RGB LED MCU

HT68FB541/HT68FB571

Revision: V1.40 Date: September 10, 2021

www.holtek.com

Table of Contents

Features

CPU Features

- Operating voltage
  - V_{DD} (MCU)
    - f_{SYS}=12MHz: 3.0V~5.5V
  - V_{DD} (USB Mode)
    - f_{SYS}=6MHz/12MHz: 3.0V~5.5V
    - f_{SYS}=16MHz: 3.3V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 12MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 109 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 4K×16~8K×16
- Data Memory: 256×8~512×8
- LED Memory: 128×8 – for HT68FB571 only
- USB Memory: 64×8
- True EEPROM Memory: 64×8
- Watchdog Timer function
- Two Comparator functions – for HT68FB541 only
- Up to 41 bidirectional I/O lines
- Dual pin-shared external interrupts
- Dual Time Base functions for generation of fixed time interrupt signals
- Two 16-bit Timer/Event Counters with PFD (Programmable Frequency Divider) functions
- Multiple 8-bit PWM outputs for LED application
- USB interface
  - USB 2.0 Full Speed compatible
  - 4 endpoints supported including endpoint 0
  - All endpoints except endpoint 0 can support interrupt and bulk transfer
  - All endpoints except endpoint 0 have a fixed FIFO size of 16, 32 and 64 bytes respectively
  - Endpoint 0 support control transfer
  - Endpoint 0 has 8 byte FIFO
  - Support 3.3V LDO and internal UDP 1.5kΩ pull-up resistor
  - Internal 12MHz RC oscillator with 0.25% accuracy for all USB modes

- Single Serial Peripheral Interface – SPI
- In Application Programming function – IAP
- In System Programming function – ISP
- Low voltage reset function
- Low voltage detect function
- Package types: 24/28-pin SSOP, 48-pin LQFP

General Description

The devices are Flash Memory LED control with USB type 8-bit high performance RISC architecture microcontrollers, designed for applications that interface directly to analog signals and require a USB interface.

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

Analog feature includes two comparators which only exist in the HT68FB541 device. Two timer/event counters are provided for external event counting, time interval/pulse width measurement or an accurate time base generation. There are also multiple 8-bit PWM outputs specially provided for multiple LED applications. Communication with the outside world is catered for by including fully integrated SPI and USB interface functions, two popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments. The external interrupts can be triggered with rising edge, falling edge or both falling and rising edges.

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

The inclusion of flexible I/O programming features along with many other features ensures that these devices will find specific excellent use in PC peripheral applications such as LED mouse, LED key pad, LED gaming keyboard, etc.

The devices are fully supported by the Holtek range of fully functional development and programming tools, providing a means for fast and efficient product development cycles.

Selection Table

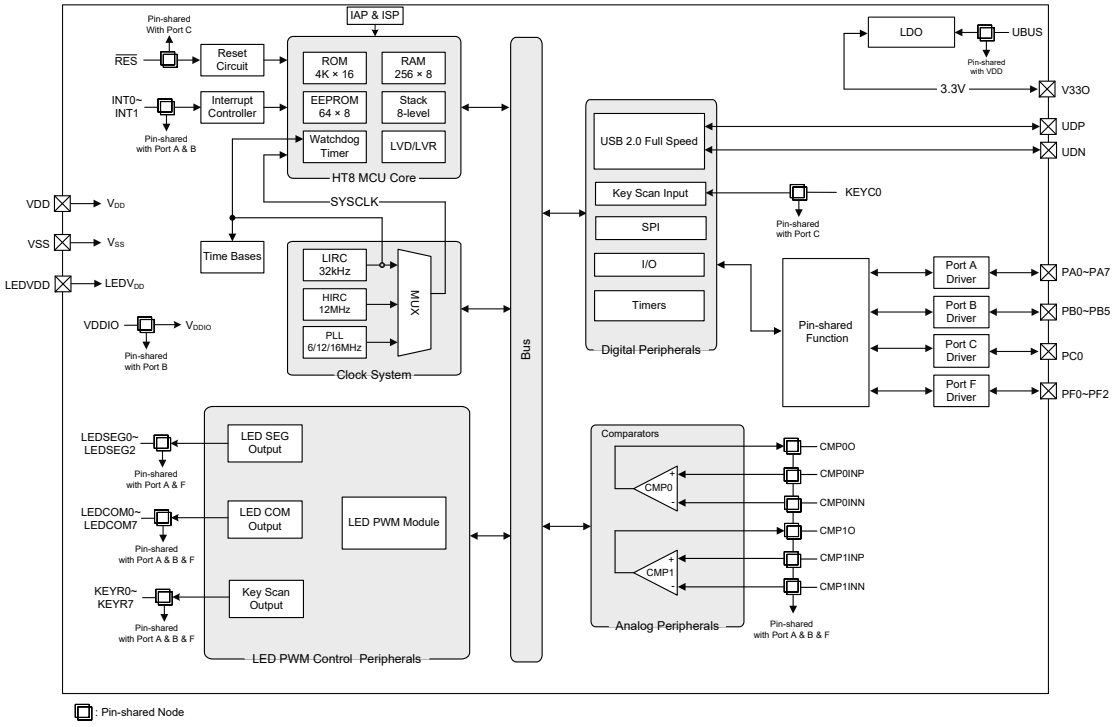
Most features are common to these devices, the main features distinguishing them are Flash Memory capacity, Data Memory capacity, I/O count, PWM count for external LEDs and package types. The following table summarises the main features of each device. As the devices exist in more than one package format, the tables reflect the situation for the package with the most pins.

Part No.	V_{DD}	Program Memory	Data Memory	LED Data Memory	Data EEPROM	I/O	IAP&ISP	External Interrupt
HT68FB541	3.0V~5.5V	4K×16	256×8	—	64×8	18	√	2
HT68FB571		8K×16	512×8	128×8		41		

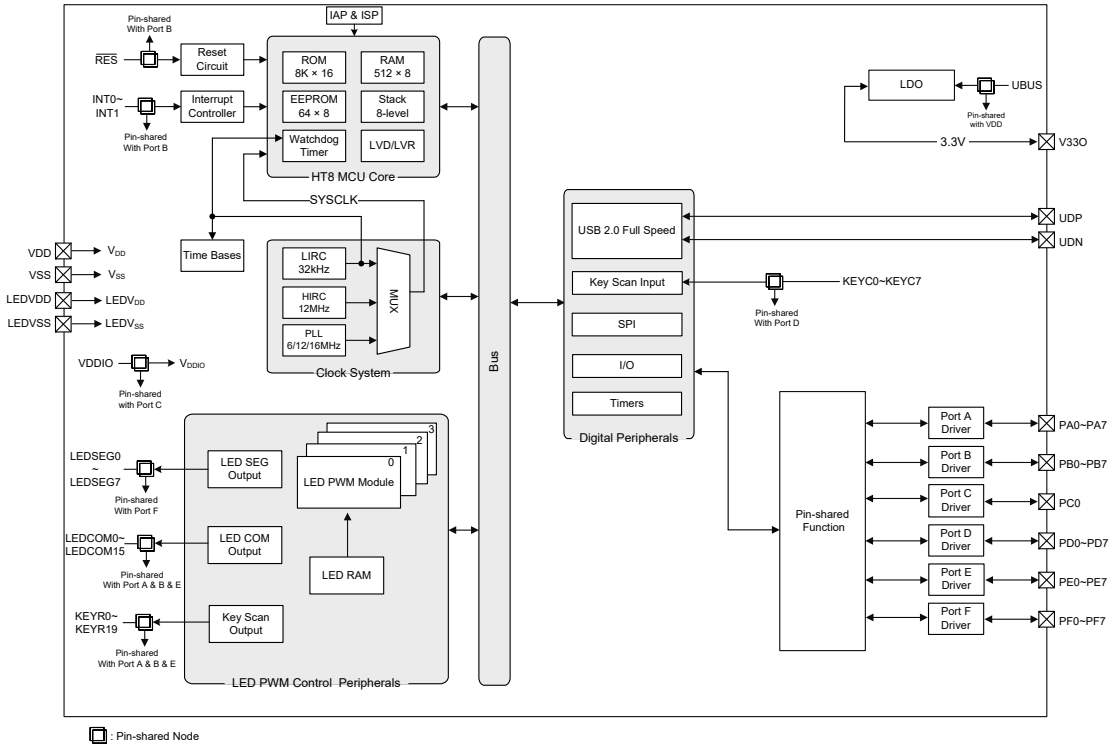
Part No.	Timer/Event Counter	Time Base	USB	SPI	LED Driver	Comparator	Stacks	Package
HT68FB541	2	2	√	√	3×8	2	8	24SSOP
HT68FB571					8×16	—		28SSOP/48LQFP

Block Diagram

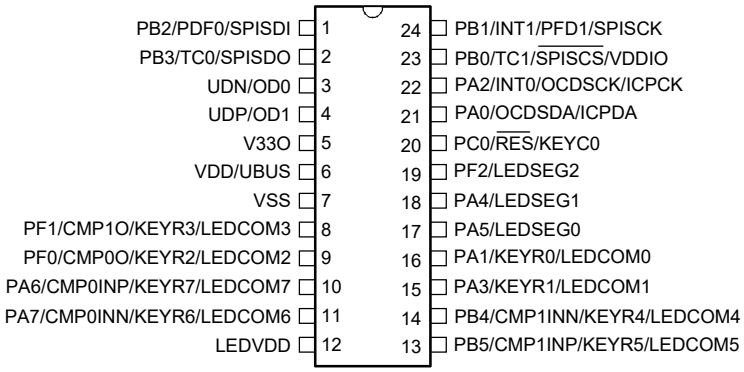
HT68FB541



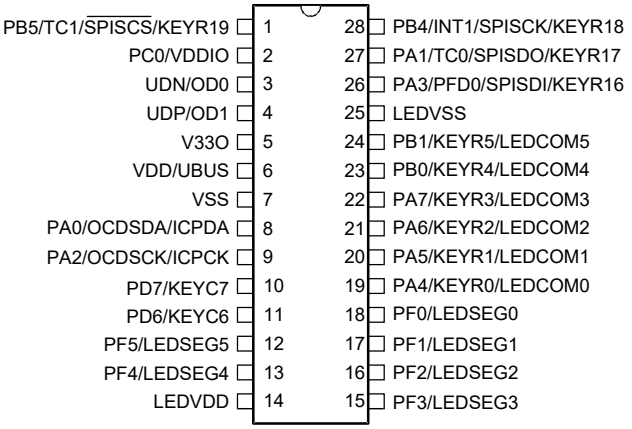
HT68FB571



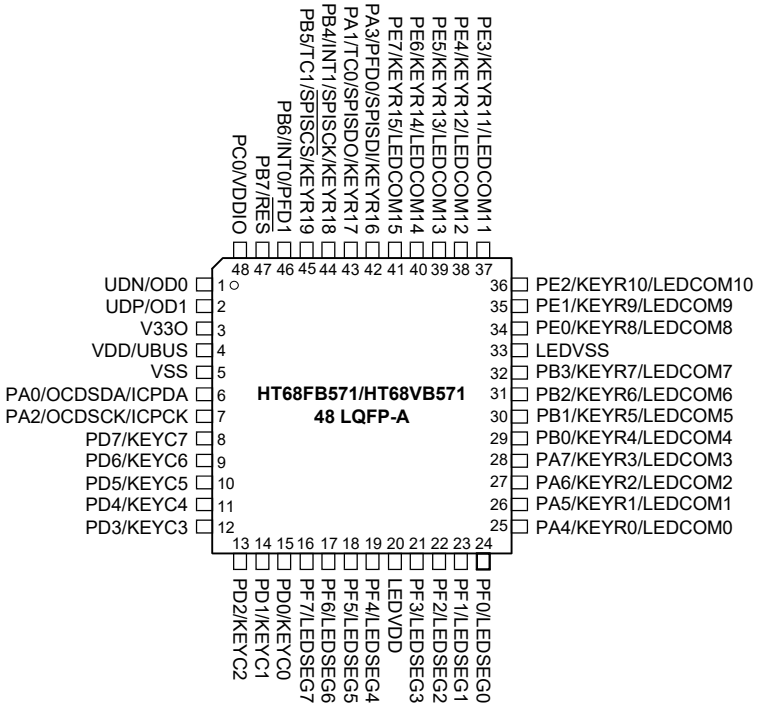
Pin Assignment



HT68FB541/HT68VB541
24 SSOP-A



HT68FB571/HT68VB571
28 SSOP-A



HT68FB571/HT68VB571
48 LQFP-A

Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.
2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the HT68VB5x1 device which is the OCDS EV chip for the HT68FB5x1 device.
3. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input condition. Refer to the "Standby Current Considerations" and "Input/Output Ports" sections.

Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet. As the Pin Description table shows the situation for the package with the most pins, not all pins in the table will be available on smaller package sizes.

HT68FB541

Pin Name	Function	OPT	I/T	O/T	Description
PA0/OCDSDA/ICPDA	PA0	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
	ICPDA	—	ST	CMOS	ICP data/address
PA1/KEYR0/LEDCOM0	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEYR0	PAS0	—	CMOS	Hardware Key Scan output pin
	LEDCOM0	PAS0	—	CMOS	LED COM output
PA2/INT0/OCDSCK/ ICPCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT0	INTEG INTC0	ST	—	External interrupt input pin 0
	OCDSCK	—	ST	—	OCDS clock, for EV chip only
	ICPCK	—	ST	—	ICP clock input
PA3/KEYR1/LEDCOM1	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEYR1	PAS0	—	CMOS	Hardware Key Scan output pin
	LEDCOM1	PAS0	—	CMOS	LED COM output
PA4/LEDSEG1	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	LEDSEG1	PAS1	—	CMOS	LED SEG output
PA5/LEDSEG0	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	LEDSEG0	PAS1	—	CMOS	LED SEG output
PA6/CMP0INP/KEYR7/ LEDCOM7	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CMP0INP	PAS1	AN	—	Positive input of Comparator 0
	KEYR7	PAS1	—	CMOS	Hardware Key Scan output pin
	LEDCOM7	PAS1	—	CMOS	LED COM output
PA7/CMP0INN/KEYR6/ LEDCOM6	PA7	PAWU PAPU PAS1	—	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CMP0INN	PAS1	AN	—	Negative input of Comparator 0
	KEYR6	PAS1	—	CMOS	Hardware Key Scan output pin
	LEDCOM6	PAS1	—	CMOS	LED COM output

Pin Name	Function	OPT	I/T	O/T	Description
PB0/TC1/$\overline{\text{SPISCS}}$/VDDIO	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TC1	PBS0	ST	—	Timer/Event Counter 1 clock input
	$\overline{\text{SPISCS}}$	PBS0	ST	CMOS	SPI slave select pin
	VDDIO	PBS0	PWR	—	PA0, PA2, PB1~PB3 external power input
PB1/INT1/PFD1/SPISCK	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT1	INTEG INTC0 PBS0	ST	—	External interrupt input pin 1
	PFD1	PBS0	—	CMOS	Programmable Frequency Divider 1
	SPISCK	PBS0	ST	CMOS	SPI serial clock
PB2/PFD0/SPISDI	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PFD0	PBS0	—	CMOS	Programmable Frequency Divider 0
	SPISDI	PBS0	ST	—	SPI serial data input
PB3/TC0/SPISDO	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TC0	PBS0	ST	—	Timer/Event Counter 0 clock input
	SPISDO	PBS0	—	CMOS	SPI serial data output
PB4/CMP1INN/KEYR4/ LEDCOM4	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CMP1INN	PBS1	AN	—	Negative input of Comparator 1
	KEYR4	PBS1	—	CMOS	Hardware Key Scan output pin
	LEDCOM4	PBS1	—	CMOS	LED COM output
PB5/CMP1INP/KEYR5/ LEDCOM5	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CMP1INP	PBS1	AN	—	Positive input of Comparator 1
	KEYR5	PBS1	—	CMOS	Hardware Key Scan output pin
	LEDCOM5	PBS1	—	CMOS	LED COM output
PC0/$\overline{\text{RES}}$/KEYC0	PC0	PCWU PCPU PCS0 RSTC	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	$\overline{\text{RES}}$	PCS0 RSTC	ST	—	Reset input pin
	KEYC0	PCS0	ST	—	Hardware Key Scan input pin
PF0/CMP0O/KEYR2/ LEDCOM2	PF0	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	CMP0O	PFS0	—	CMOS	Comparator 0 output
	KEYR2	PFS0	—	CMOS	Hardware Key Scan output pin
	LEDCOM2	PFS0	—	CMOS	LED COM output
PF1/CMP1O/KEYR3/ LEDCOM3	PF1	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	CMP1O	PFS0	—	CMOS	Comparator 1 output
	KEYR3	PFS0	—	CMOS	Hardware Key Scan output pin
	LEDCOM3	PFS0	—	CMOS	LED COM output
PF2/LEDSEG2	PF2	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	LEDSEG2	PFS0	—	CMOS	LED SEG output
V33O	V33O	—	—	PWR	USB 3.3V regulator output

Pin Name	Function	OPT	I/T	O/T	Description
UDN/OD0	UDN	—	ST	CMOS	USB D- line
	OD0	—	ST	NMOS	NMOS Open Drain I/O pin
UDP/OD1	UDP	—	ST	CMOS	USB D+ line
	OD1	—	ST	NMOS	NMOS Open Drain I/O pin
VDD/UBUS	VDD	—	PWR	—	Digital positive power supply
	UBUS	—	PWR	—	USB positive power supply
VSS	VSS	—	PWR	—	Digital negative power supply
LEDVDD	LEDVDD	—	PWR	—	LEDSEG/LEDCOM positive power supply

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

HT68FB571

Pin Name	Function	OPT	I/T	O/T	Description
PA0/OCDSDA/ICPDA	PA0	PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
	ICPDA	—	ST	CMOS	ICP data/address
PA1/TC0/SPISDO/ KEYR17	PA1	PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TC0	PAS0	ST	—	Timer/Event Counter 0 clock input
	SPISDO	PAS0	—	CMOS	SPI serial data output
	KEYR17	PAS0	—	CMOS	Hardware Key Scan output pin
PA2/OCDSCK/ICPCK	PA2	PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	OCDSCK	—	ST	—	OCDS clock, for EV chip only
	ICPCK	—	ST	—	ICP clock input
PA3/PFD0/SPISDI/ KEYR16	PA3	PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PFD0	PAS0	—	CMOS	Programmable Frequency Divider 0
	SPISDI	PAS0	ST	—	SPI serial data input
	KEYR16	PAS0	—	CMOS	Hardware Key Scan output pin
PA4/KEYR0/LEDCOM0	PA4	PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR0	PAS1	—	CMOS	Hardware Key Scan output pin
	LEDCOM0	PAS1	—	CMOS	LED COM output
PA5/KEYR1/LEDCOM1	PA5	PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR1	PAS1	—	CMOS	Hardware Key Scan output pin
	LEDCOM1	PAS1	—	CMOS	LED COM output
PA6/KEYR2/LEDCOM2	PA6	PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR2	PAS1	—	CMOS	Hardware Key Scan output pin
	LEDCOM2	PAS1	—	CMOS	LED COM output
PA7/KEYR3/LEDCOM3	PA7	PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR3	PAS1	—	CMOS	Hardware Key Scan output pin
	LEDCOM3	PAS1	—	CMOS	LED COM output

Pin Name	Function	OPT	I/T	O/T	Description
PB0/KEYR4/LEDCOM4	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR4	PBS0	—	CMOS	Hardware Key Scan output pin
	LEDCOM4	PBS0	—	CMOS	LED COM output
PB1/KEYR5/LEDCOM5	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR5	PBS0	—	CMOS	Hardware Key Scan output pin
	LEDCOM5	PBS0	—	CMOS	LED COM output
PB2/KEYR6/LEDCOM6	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR6	PBS0	—	CMOS	Hardware Key Scan output pin
	LEDCOM6	PBS0	—	CMOS	LED COM output
PB3/KEYR7/LEDCOM7	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR7	PBS0	—	CMOS	Hardware Key Scan output pin
	LEDCOM7	PBS0	—	CMOS	LED COM output
PB4/INT1/SPISCK/ KEYR18	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT1	INTEG INTC0 PBS1	ST	—	External interrupt input pin
	SPISCK	PBS1	ST	CMOS	SPI serial clock
	KEYR18	PBS1	—	CMOS	Hardware Key Scan output pin
PB5/TC1/$\overline{\text{SPISCS}}$/ KEYR19	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	TC1	PBS1	ST	—	Timer/Event Counter 1 clock input
	$\overline{\text{SPISCS}}$	PBS1	ST	CMOS	SPI slave select pin
	KEYR19	PBS1	—	CMOS	Hardware Key Scan output pin
PB6/INT0/PFD1	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT0	INTEG INTC0 PBS1	ST	—	External interrupt input pin
	PFD1	PBS1	—	CMOS	Programmable Frequency Divider 1
PB7/$\overline{\text{RES}}$	PB7	PBPU RSTC	ST	CMOS	General purpose I/O. Register enabled pull-up
	$\overline{\text{RES}}$	RSTC	ST	—	Reset input pin
PC0/VDDIO	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	VDDIO	PCS0	PWR	—	PA1, PA3, PB4~PB7 external power input
PD0/KEYC0	PD0	PDWU PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEYC0	PDS0	ST	—	Hardware Key Scan input pin
PD1/KEYC1	PD1	PDWU PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEYC1	PDS0	ST	—	Hardware Key Scan input pin
PD2/KEYC2	PD2	PDWU PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEYC2	PDS0	ST	—	Hardware Key Scan input pin

Pin Name	Function	OPT	I/T	O/T	Description
PD3/KEYC3	PD3	PDWU PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEYC3	PDS0	ST	—	Hardware Key Scan input pin
PD4/KEYC4	PD4	PDWU PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEYC4	PDS1	ST	—	Hardware Key Scan input pin
PD5/KEYC5	PD5	PDWU PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEYC5	PDS1	ST	—	Hardware Key Scan input pin
PD6/KEYC6	PD6	PDWU PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEYC6	PDS1	ST	—	Hardware Key Scan input pin
PD7/KEYC7	PD7	PDWU PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEYC7	PDS1	ST	—	Hardware Key Scan input pin
PE0/KEYR8/LEDCOM8	PE0	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR8	PES0	—	CMOS	Hardware Key Scan output pin
	LEDCOM8	PES0	—	CMOS	LED COM output
PE1/KEYR9/LEDCOM9	PE1	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR9	PES0	—	CMOS	Hardware Key Scan output pin
	LEDCOM9	PES0	—	CMOS	LED COM output
PE2/KEYR10/ LEDCOM10	PE2	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR10	PES0	—	CMOS	Hardware Key Scan output pin
	LEDCOM10	PES0	—	CMOS	LED COM output
PE3/KEYR11/ LEDCOM11	PE3	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR11	PES0	—	CMOS	Hardware Key Scan output pin
	LEDCOM11	PES0	—	CMOS	LED COM output
PE4/KEYR12/ LEDCOM12	PE4	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR12	PES1	—	CMOS	Hardware Key Scan output pin
	LEDCOM12	PES1	—	CMOS	LED COM output
PE5/KEYR13/ LEDCOM13	PE5	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR13	PES1	—	CMOS	Hardware Key Scan output pin
	LEDCOM13	PES1	—	CMOS	LED COM output
PE6/KEYR14/ LEDCOM14	PE6	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR14	PES1	—	CMOS	Hardware Key Scan output pin
	LEDCOM14	PES1	—	CMOS	LED COM output
PE7/KEYR15/ LEDCOM15	PE7	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEYR15	PES1	—	CMOS	Hardware Key Scan output pin
	LEDCOM15	PES1	—	CMOS	LED COM output

Pin Name	Function	OPT	I/T	O/T	Description
PF0/LEDSEG0	PF0	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	LEDSEG0	PFS0	—	CMOS	LED SEG output
PF1/LEDSEG1	PF1	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	LEDSEG1	PFS0	—	CMOS	LED SEG output
PF2/LEDSEG2	PF2	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	LEDSEG2	PFS0	—	CMOS	LED SEG output
PF3/LEDSEG3	PF3	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	LEDSEG3	PFS0	—	CMOS	LED SEG output
PF4/LEDSEG4	PF4	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	LEDSEG4	PFS1	—	CMOS	LED SEG output
PF5/LEDSEG5	PF5	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	LEDSEG5	PFS1	—	CMOS	LED SEG output
PF6/LEDSEG6	PF6	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	LEDSEG6	PFS1	—	CMOS	LED SEG output
PF7/LEDSEG7	PF7	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	LEDSEG7	PFS1	—	CMOS	LED SEG output
V33O	V33O	—	—	PWR	USB 3.3V regulator output
UDN/OD0	UDN	—	ST	CMOS	USB D- line
	OD0	—	ST	NMOS	NMOS Open Drain I/O pin
UDP/OD1	UDP	—	ST	CMOS	USB D+ line
	OD1	—	ST	NMOS	NMOS Open Drain I/O pin
VDD/UBUS	VDD	—	PWR	—	Digital Positive Power supply
	UBUS	—	PWR	—	USB positive power supply
VSS	VSS	—	PWR	—	Digital Negative Power supply
LEDVDD	LEDVDD	—	PWR	—	LEDSEG/LEDCOM Positive Power supply
LEDVSS	LEDVSS	—	PWR	—	LEDSEG/LEDCOM Negative Power supply, ground

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

Absolute Maximum Ratings

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

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

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

Operating Temperature..... -40°C to 85°C

I_{OH} Total .....-200mA (for HT68FB541)

I_{OH} Total .....-100mA (for HT68FB571)

I_{OL} Total .....150mA (for HT68FB541)

I_{OL} Total .....200mA (for HT68FB571)

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 device reliability.

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (HIRC)	—	$f_{SYS}=f_{HIRC}$=12MHz	3.0	—	5.5	V
	Operating Voltage (LIRC)	—	$f_{SYS}=f_{LIRC}$=32kHz	3.0	—	5.5	
	Operating Voltage (PLL)	—	PLLEN=1, $f_{SYS}=f_{PLL}$=6MHz	3.0	—	5.5	
		—	PLLEN=1, $f_{SYS}=f_{PLL}$=12MHz	3.0	—	5.5	
		—	PLLEN=1, $f_{SYS}=f_{PLL}$=16MHz	3.3	—	5.5	

Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	Operating Current (HIRC)	3V	$f_{SYS}=f_{HIRC}$=12MHz	—	1.6	3.0	mA
		5V		—	2.8	6.0	
	Operating Current (LIRC)	3V	$f_{SYS}=f_{LIRC}$=32kHz	—	16	30	μA
		5V		—	28	50	
	Operating Current (PLL)	3V	PLLEN=1, $f_{SYS}=f_{PLL}$=6MHz	—	2	3	mA
		5V		—	3	4	
		3V	PLLEN=1, $f_{SYS}=f_{PLL}$=12MHz	—	2.1	3.5	
		5V		—	3.8	7.0	
		3.3V	PLLEN=1, $f_{SYS}=f_{PLL}$=16MHz	—	3.2	6.5	
		5V		—	4.5	9.0	

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

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

Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max	Max.@ 85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	3V	WDT off	—	0.11	0.15	2.00	μA
		5V		—	0.18	0.38	2.90	
		3V	WDT on	—	—	3	3	
		5V		—	—	5	5	
	IDLE0 Mode (LIRC)	3V	f_{SUB} on, $f_{SYS}=f_{HIRC}$=12MHz	—	1.5	3.0	3.5	μA
		5V		—	2.8	5.0	5.5	
		3V	f_{SUB} on, $f_{SYS}=f_{SUB}$	—	3	5	5	
		5V		—	5	10	10	
	IDLE1 Mode (HIRC)	3V	f_{SUB} on, $f_{SYS}=f_{HIRC}$=12MHz	—	0.65	1.40	1.40	mA
		5V		—	1.3	3.0	3.0	
	IDLE1 Mode (PLL)	3V	f_{SUB} on, PLLEN=1 $f_{SYS}=f_{PLL}$=6MHz	—	1.0	2.5	2.5	mA
		5V		—	2.0	3.5	3.5	
		3V	f_{SUB} on, PLLEN=1 $f_{SYS}=f_{PLL}$=12MHz	—	1.5	3.0	3.0	
		5V		—	2.5	4.0	4.0	
		3.3V	f_{SUB} on, PLLEN=1 $f_{SYS}=f_{PLL}$=16MHz	—	1.7	3.5	3.5	
		5V		—	3	5	5	

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

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

A.C. Characteristics

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

High Speed Internal Oscillator – HIRC – Frequency Accuracy

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{HIRC}	12MHz Writer Trimmed HIRC frequency	4.4V~5.25V	USB mode and CLKADJ=1, UDP/UDN plug in the HOST Ta=-40°C~85°C	-0.25%	12	+0.25%	MHz
		5V	Ta=25°C	-2%	12	+2%	
		5V	Ta=0°C~70°C	-3%	12	+3%	
		5V	Ta=-40°C~85°C	-3.5%	12	+3.5%	
		3.0V~5.5V	Ta=0°C~70°C	-4%	12	+4%	
		3.0V~5.5V	Ta=-40°C~85°C	-4.5%	12	+4.5%	

Note: 1. The 5V value for V_{DD} is provided as this is the fixed voltage at which the HIRC frequency is trimmed by the writer.

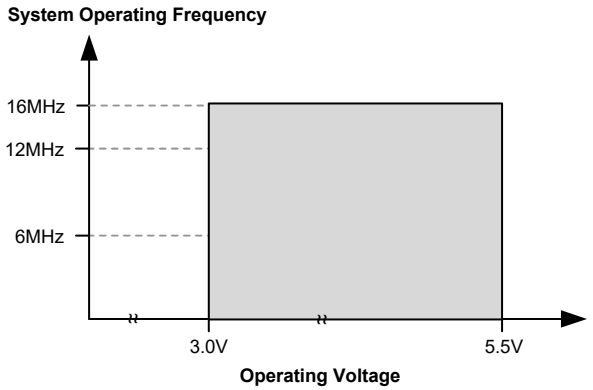
2. The row below the 5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 5V for application voltage ranges from 3.0V to 5.5V.

3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

Low Speed Internal Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{LIRC}	LIRC Frequency	3.0V~5.5V	-40°C~85°C	-7%	32	+7%	kHz
t_{START}	LIRC Start up Time	—	-40°C~85°C	—	—	100	μs

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Timer Period (Wake-up from IDLE/SLEEP Mode and f_{SYS} Off, $\overline{RES}$ Pin Reset)	—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{SYS}
		—	f_{PLL} off → on (PLLF=1)	—	2560	—	$t_{PLL(48MHz)}$
		—	$f_{SYS}=f_{LIRC}$	—	2	—	t_{SYS}
	System Start-up Timer Period (SLOW mode ↔ NORMAL Mode)	—	f_{HIRC} off → on (HIRCF=1)	—	16	—	t_{HIRC}
		—	f_{PLL} off → on (PLLF=1)	—	2560	—	$t_{PLL(48MHz)}$
	System start-up timer period (wake-up from IDLE/SLEEP Mode, f_{SYS} on)	—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$ or f_{PLL}	—	2	—	t_{SYS}
		—	$f_{SYS}=f_{LIRC}$	—	2	—	t_{SYS}
t_{RSTD}	System Reset Delay Time Reset Source from Power-on Reset or LVR Hardware Reset	—	RR_{POR}=5 V/ms	42	48	54	ms
	System Reset Delay Time LVRC/WDTC/RSTC Software Reset	—	—				
	System Reset Delay Time Reset Source from WDT Overflow or $\overline{RES}$ Pin Reset	—	—	14	16	40	ms
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

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

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

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

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

Input/Output Characteristics

Ta=-40°C~85°C

Input/Output (without Multi-power) D.C. Characteristics – HT68FB541

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports (Except $\overline{RES}$ Pin)	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
	Input Low Voltage for $\overline{RES}$ Pin	—	—	0	—	$0.4V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports (Except $\overline{RES}$ Pin)	5V	—	3.5	—	5	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
	Input High Voltage for $\overline{RES}$ Pin	—	—	$0.9V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Pins	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Pins (Except PF0~PF1, PA6~PA7, PA1, PA3, PB4~PB5)	3V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	
	Source Current for I/O Pins (PF0~PF1, PA6~PA7, PA1, PA3, PB4~PB5)	3V	$V_{OH}=0.9V_{DD}$, SLEDC[m+1, m]=00B (n=0, 1; m=0, 2, 4, 6)	-7	-15	—	mA
		5V		-15	-33	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDC[m+1, m]=01B (n=0, 1; m=0, 2, 4, 6)	-15	-33	—	
		5V		-33	-67	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDC[m+1, m]=10B (n=0, 1; m=0, 2, 4, 6)	-33	-67	—	
		5V		-67	-100	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDC[m+1, m]=11B (n=0, 1; m=0, 2, 4, 6)	-67	-120	—	
		5V		-120	-135	—	
R_{PH}	Pull-high Resistance for I/O Ports Note	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I_{LEAK}	Input Leakage Current	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{INT}	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs
t_{RES}	External Reset Minimum Low Pulse Width	—	—	10	—	—	μs
t_{TC}	TCn Input Pin Minimum Pulse Width	—	—	25	—	—	ns

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

Input/Output (with Multi-power) D.C. Characteristics – HT68FB541

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	V_{DD} Power Supply for Multi-power I/O Pins	—	—	3.0	5.0	5.5	V
V_{DDIO}	V_{DDIO} Power Supply for Multi-power I/O Pins	—	—	3.0	—	V_{DD}	V
V_{V33O}	V_{V33O} Power Supply for Multi-power I/O Pins	4.0V~5.5V	V33O output enable	3.0	3.3	3.6	V
V_{IL}	Input Low Voltage for Multi-power I/O Pins	5V	Pin power=V_{DD} or V_{DDIO}, V_{DDIO}=V_{DD}	0.0	—	1.5	V
		—	Pin power=V_{DD} or V_{DDIO}	0.0	—	0.2 (V_{DD}/V_{DDIO})	
V_{IH}	Input High Voltage for Multi-power I/O Pins	5V	Pin power=V_{DD} or V_{DDIO}, V_{DDIO}=V_{DD}	3.5	—	5.0	V
		—	Pin power=V_{DD} or V_{DDIO}	0.8 (V_{DD}/V_{DDIO})	—	V_{DD}/V_{DDIO}	
I_{OL}	Sink Current for Multi-power I/O Pins	3V	V_{OL}=0.1(V_{DD}/V_{DDIO}) V_{DDIO}=V_{DD}	16	32	—	mA
		5V	V_{OL}=0.1(V_{DD}/V_{DDIO}) V_{DDIO}=V_{DD}	32	65	—	
			V_{OL}=0.1V_{DDIO}, V_{DDIO}=3V	20	40	—	
I_{OH}	Source Current for Multi-power I/O Pins	3V	V_{OH}=0.9(V_{DD}/V_{DDIO}) V_{DDIO}=V_{DD}	-4	-8	—	mA
		5V	V_{OH}=0.9(V_{DD}/V_{DDIO}) V_{DDIO}=V_{DD}	-8	-16	—	
			V_{OH}=0.9V_{DDIO}, V_{DDIO}=3V	-2.5	-5	—	
R_{PH}	Pull-high Resistance for Multi-power I/O Pins (2)	3V	Pin power=V_{DD} or V_{DDIO} V_{DDIO}=V_{DD}	20	60	100	kΩ
		5V	Pin power=V_{DD} or V_{DDIO} V_{DDIO}=V_{DD}	10	30	50	
			V_{DDIO}=3V	36	110	180	
I_{LEAK}	Input Leakage Current for Multi-power I/O Pins	5V	V_{IN}=V_{SS} or V_{IN}=V_{DD} or V_{DDIO}	—	—	±1	μA

Note: 1. The multi-power pins are PA0, PA2, PB1~PB3 pins.

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

Input/Output (without Multi-power) D.C. Characteristics – HT68FB571

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports (Except PD0~PD7 & $\overline{RES}$ Pin)	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
	Input Low Voltage for I/O Ports (PD0~PD7)	5V	INHLVC[m]=0 (m=0, 1)	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	
		5V	INHLVC[m]=1 (m=0, 1)	0	—	0.8	
	Input Low Voltage for $\overline{RES}$ Pin	—	—	0	—	0.4 V_{DD}	V
V_{IH}	Input High Voltage for I/O Ports (Except PD0~PD7& $\overline{RES}$ Pin)	5V	—	3.5	—	5	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
	Input High Voltage for I/O Ports (PD0~PD7)	5V	INHLVC[m]=0 (m=0, 1)	3.5	—	5	V
		—		$0.8V_{DD}$	—	V_{DD}	
		5V	INHLVC[m]=1 (m=0, 1)	2.5	—	5	
	Input High Voltage for $\overline{RES}$ Pin	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	Sink Current for I/O Pins (Except PE0~PE7, PB0~PB3, PA4~PA7, PF0~PF7)	3V	$V_{OL}=0.1V_{DD}$	5	10	—	mA
		5V		10	20	—	
	Sink Current for I/O Pins (PE0~PE7, PB0~PB3, PA4~PA7)	3V	$V_{OL}=0.1V_{DD}$	20	40	—	mA
		5V		40	80	—	
	Sink Current for I/O Pins (PF0~PF7)	3V	$V_{OL}=0.1V_{DD}$	2.5	5	—	mA
		5V		5	10	—	
I_{OH}	Source Current for I/O Pins (Except PF0~PF7)	3V	$V_{OH}=0.9V_{DD}$	-2	-5	—	mA
		5V		-5	-10	—	
	Source Current for I/O Pins (PF0~PF7)	3V	$V_{OH}=0.9V_{DD}$, SLEDC[m+1, m]=00B (m=0, 2)	-0.9	-1.8	—	mA
		5V		-1.8	-3.6	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDC[m+1, m]=01B (m=0, 2)	-1.6	-3.2	—	
		5V		-3.2	-6.4	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDC[m+1, m]=10B (m=0, 2)	-2.3	-4.5	—	
		5V		-4.5	-9.1	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDC[m+1, m]=11B (m=0, 2)	-5	-10	—	
		5V		-10	-20	—	
R_{PH}	Pull-high Resistance for I/O Ports Note	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I_{LEAK}	Input Leakage Current	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{INT}	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs
t_{RES}	External Reset Minimum Low Pulse Width	—	—	10	—	—	μs
t_{TC}	TCn Input Pin Minimum Pulse Width	—	—	25	—	—	ns

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

Input/Output (with Multi-power) D.C. Characteristics – HT68FB571

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	V_{DD} Power Supply for Multi-power I/O Pins	—	—	3.0	5	5.5	V
V_{DDIO}	V_{DDIO} Power Supply for Multi-power I/O Pins	—	—	3.0	—	V_{DD}	V
V_{V33O}	V_{V33O} Power Supply for Multi-power I/O Pins	4.0V ~5.5V	V33O output enable	3.0	3.3	3.6	V
V_{IL}	Input Low Voltage for Multi-power I/O Pins	5V	Pin power=V_{DD} or V_{DDIO} V_{DDIO}=V_{DD}	0	—	1.5	V
		—	Pin power=V_{DD} or V_{DDIO}	0	—	0.2 (V_{DD}/V_{DDIO})	
V_{IH}	Input High Voltage for Multi-power I/O Pins	5V	Pin power=V_{DD} or V_{DDIO} V_{DDIO}=V_{DD}	3.5	—	5	V
		—	Pin power=V_{DD} or V_{DDIO}	0.8 (V_{DD}/V_{DDIO})	—	V_{DD}/V_{DDIO}	
I_{OL}	Sink Current for Multi-power I/O Pins	3V	V_{OL}=0.1(V_{DD}/V_{DDIO}) V_{DDIO}=V_{DD}	5	10	—	mA
		5V	V_{OL}=0.1(V_{DD}/V_{DDIO}) V_{DDIO}=V_{DD}	10	20	—	
			V_{OL}=0.1V_{DDIO}, V_{DDIO}=3V	7.5	15.0	—	
I_{OH}	Source Current for Multi-power I/O Pins	3V	V_{OH}=0.9(V_{DD}/V_{DDIO}) V_{DDIO}=V_{DD}	-2	-5	—	mA
		5V	V_{OH}=0.9(V_{DD}/V_{DDIO}) V_{DDIO}=V_{DD}	-5	-10	—	
			V_{OH}=0.9V_{DDIO} V_{DDIO}=3V	-1.5	-3.0	—	
R_{PH}	Pull-high Resistance for Multi-power I/O Pins (2)	3V	Pin power=V_{DD} or V_{DDIO} V_{DDIO}=V_{DD}	20	60	100	kΩ
		5V	Pin power=V_{DD} or V_{DDIO} V_{DDIO}=V_{DD}	10	30	50	
			V_{DDIO}=3V	36	110	180	
I_{LEAK}	Input Leakage Current for Multi-power I/O Pins	5V	V_{IN}=V_{SS} or V_{IN}=V_{DD} or V_{DDIO}	—	—	±1	μA

Note: 1. The multi-power pins are PA1, PA3, PB4~PB7 pins.

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

Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{RW}	V_{DD} for Read / Write	—	—	V_{DDmin}	—	V_{DDmax}	V
Flash Program Memory / Data EEPROM Memory							
t_{DEW}	Erase / Write Cycle Time – Flash Program Memory	—	—	—	2	3	ms
	Write Cycle Time – Data EEPROM Memory	—	—	—	4	6	
I_{DDPGM}	Programming / Erase Current on V_{DD}	—	—	—	—	5.0	mA
E_P	Cell Endurance – Flash Program Memory	—	—	10K	—	—	E/W
	Cell Endurance – Data EEPROM Memory	—	—	100K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

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

LVR/LVD Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	-5%	2.1	+5%	V
			LVR enable, voltage select 2.55V		2.55		
			LVR enable, voltage select 3.15V		3.15		
			LVR enable, voltage select 3.8V		3.8		
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 2.0V	-5%	2.0	+5%	V
			LVD enable, voltage select 2.2V		2.2		
			LVD enable, voltage select 2.4V		2.4		
			LVD enable, voltage select 2.7V		2.7		
			LVD enable, voltage select 3.0V		3.0		
			LVD enable, voltage select 3.3V		3.3		
			LVD enable, voltage select 3.6V		3.6		
			LVD enable, voltage select 4.0V		4.0		
$I_{LVRLVDBG}$	Operating Current	3V	LVD enable, LVR enable, VBGEN=0	—	—	20	μA
		5V		—	20	25	
		3V	LVD enable, LVR enable, VBGEN=1	—	—	25	μA
		5V		—	25	30	
t_{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	18	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs
I_{LVR}	Additional Current for LVR Enable	—	LVD disable, VBGEN=0	—	—	24	μA

USB Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	Operating Current (USB)	3V	$f_{SYS}=f_{PLL}$=6MHz, No load, USB and PLL on, other peripherals off	—	4.5	9.0	mA
		5V		—	9.5	15.0	
		3V	$f_{SYS}=f_{PLL}$=12MHz, No load, USB and PLL on, other peripherals off	—	5	10	mA
		5V		—	10.5	16.0	
		5V	$f_{SYS}=f_{PLL}$=16MHz, No load, USB and PLL on, other peripherals off	—	11	18	mA
I_{SUS}	Suspend Current (USB) (IDLE0 Mode)	5V	f_H off, $f_{SUB}=f_{LIRC}$ on, No load, USB and PLL on, other peripherals off, SUSP2=0, RCTRL=0	—	360	450	μA
		5V	f_H off, $f_{SUB}=f_{LIRC}$ on, No load, USB and PLL on, other peripherals off, SUSP2=1, RCTRL=1	—	240	330	μA
V_{V33O}	3.3V Regulator Output Voltage	5V	I_{V33O}=70mA	3.0	3.3	3.6	V
R_{UDP}	Pull-high Resistance of UDP to V33O	3.3V	—	-5%	1.5	+5%	kΩ
	Pull-high Resistance of UDP to UBUS	5V	RCTRL=1	4.9	7.8	12.0	kΩ
R_{UDPN}	Pull-high Resistance of UDP/UDN to UBUS	5V	PU=1	300	650	1000	kΩ
R_{PL}	Pull-low Resistance of UBUS	5V	SUSP2=1, RUBUS=0	0.5	1.0	1.5	MΩ
R_{OD}	Pull-high Resistance of OD0/OD1 to VDD	5V	UMS[2:0]=001B	2.0	4.7	8.0	kΩ
I_{OL_OD}	Sink Current of OD0/OD1	5V	UMS[2:0]=001B, V_{OL}=0.1V_{DD}	8	12	—	mA
V_{IH}	Input High Voltage of OD0/OD1	5V	UMS[2:0]=001B, OD1O/OD0O=11B	2	—	5	V
V_{IL}	Input Low Voltage of OD0/OD1	5V	UMS[2:0]=001B, OD1O/OD0O=11B	0	—	0.8	V

Note: For I_{SUS} and R_{UDP}, a 15kΩ resistor should be connected between UDP/UDN pin and GND.

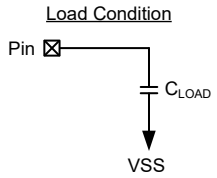
Comparator Characteristics – HT68FB541 Only

Ta=-40°C~85°C
All measurement is under CMPnINP input voltage=(V_{DD}-1.4)/2 and remain constant

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{RP}	Response Time	3.0V~5.5V	With 10mV overdrive (2)	—	—	3	μs
		3V	With 100mV overdrive (1) (2)	—	200	700	ns
		5V		—	200	400	

Note: 1. Measured with comparator one input pin at V_{CM}=(V_{DD}-1.4)/2 while the other pin input transition from V_{SS} to (V_{CM}+100mV) or from V_{DD} to (V_{CM}-100mV).

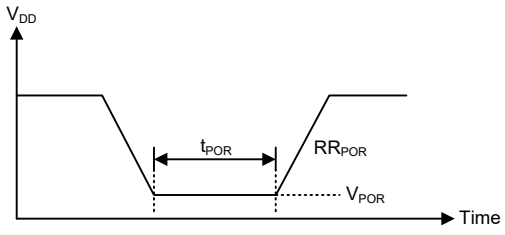
2. Load Condition: C_{LOAD}=50pF.



Power-on Reset Characteristics

Ta=-40°C~85°C

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



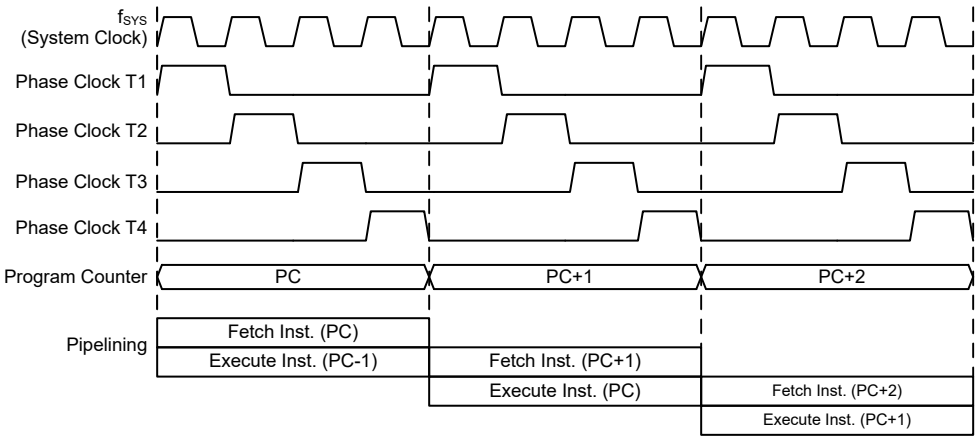
System Architecture

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

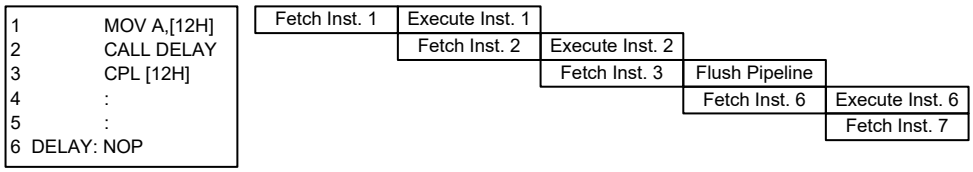
Clocking and Pipelining

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

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



System Clocking and Pipelining



Instruction Fetching

Program Counter

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

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

Device	Program Counter	
	Program Counter High Byte	PCL Register
HT68FB541	PC11~PC8	PCL7~PCL0
HT68FB571	PC12~PC8	PCL7~PCL0

Program Counter

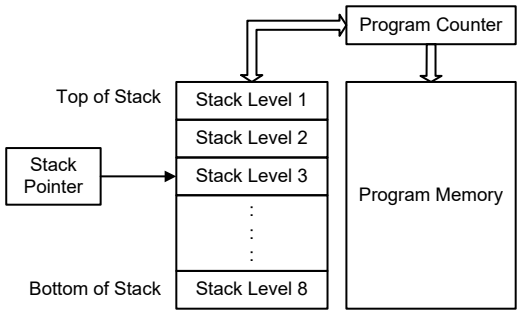
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 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,
  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
  LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
  LRRA, LRR, LRRCA, LRRC, LRLA, LRL, LRLCA, LRLC
- Increment and Decrement:
  INCA, INC, DECA, DEC,
  LINCA, LINC, LDECA, LDEC
- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI,
  LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

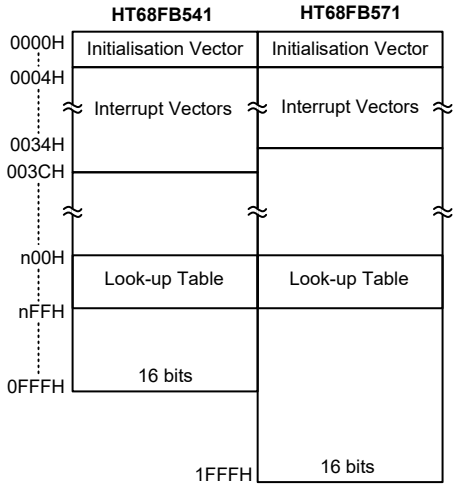
Flash Program Memory

The Program Memory is the location where the user code or program is stored. For these devices 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, these Flash devices offer users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

Device	Capacity
HT68FB541	4K×16
HT68FB571	8K×16

Structure

The Program Memory has a capacity of 4K×16 to 8K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



Program Memory Structure

Special Vectors

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

Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the corresponding table read instruction such as “TABRD [m]” when the memory [m] is located in sector 0. If the memory [m] is located in other sectors, the data can be retrieved from the program memory using the corresponding extended table read instruction such as “LTABRD [m]”. When the

instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

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

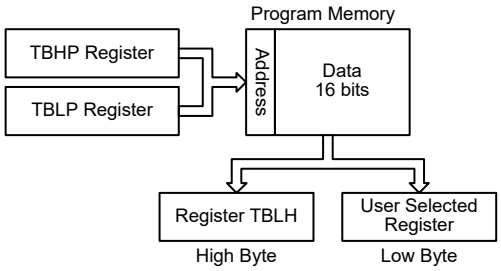


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement.

The value at this ORG statement is "1F00H" which refers to the start address of the last page within the 8K words Program Memory of the HT68FB571 device if the ISP bootloader provided by Holtek IDE tool is not used. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "1F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specified address pointed by the TBLP and TBHP registers if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

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

Table Read Program Example – HT68FB571

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

```
tabrd tempreg2     ; transfers value in table referenced by table pointer
                   ; data at program memory address "1F05H" transferred to
                   ; tempreg2 and TBLH in this example the data "1AH" is
                   ; transferred to tempreg1 and data "0FH" to tempreg2
:
:
org 1F00h          ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

In Circuit Programming – ICP

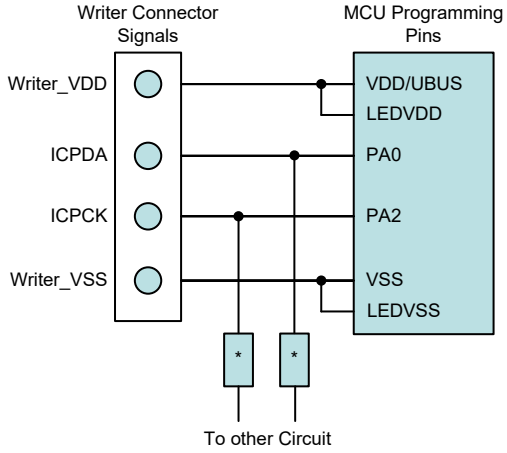
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VDD	VDD/UBUS & LEDVDD	Power Supply
VSS	VSS & LEDVSS	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 is beyond the scope of this document and will be supplied in supplementary literature.

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



Note: 1. * may be resistor or capacitor. The resistance of * must be greater than 1kΩ or the capacitance of * must be less than 1nF.
2. The LEDVSS pin only exists in the HT68FB571 device.

On-Chip Debug Support – OCDS

There is an EV chip named HT68VB5x1 which is used to emulate the HT68FB5x1 device. The EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the 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 device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

Note: The LEDVSS pin only exists in the HT68FB571 device.

In Application Programming – IAP

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

Flash Memory Read/Write Size

The Flash memory Erase and Write operations are carried out in a page format while the Read operation is carried out in a word format. The page size and write buffer size are both assigned with a capacity of 32 words. Note that the Erase operation should be executed before the Write operation is executed.

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

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

Operations	Format
Erase	32 words/page
Write	32 words/time
Read	1 word/time
Note: Page size=Write buffer size=32 words.	

IAP Operation Format

Erase Page	FARH	FARL [7:5]	FARL [4:0]
0	0000 0000	000	x xxxx
1	0000 0000	001	x xxxx
2	0000 0000	010	x xxxx
3	0000 0000	011	x xxxx
4	0000 0000	100	x xxxx
:	:	:	:
126	0000 1111	110	x xxxx
127	0000 1111	111	x xxxx

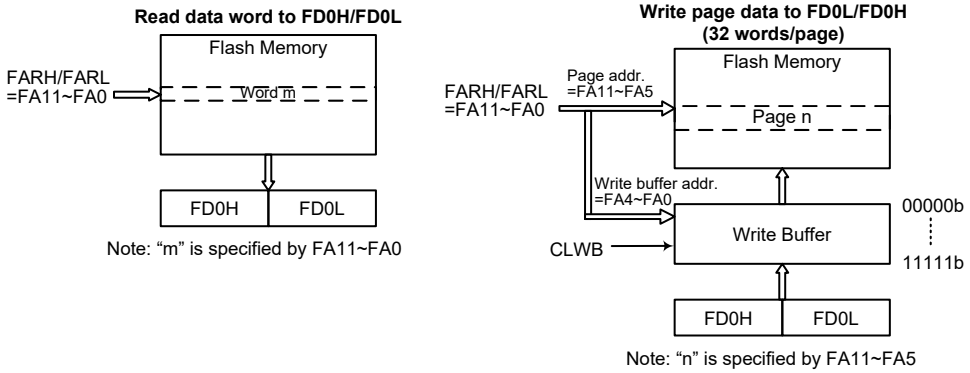
"x": Don't care

Erase Page Number and Selection – HT68FB541

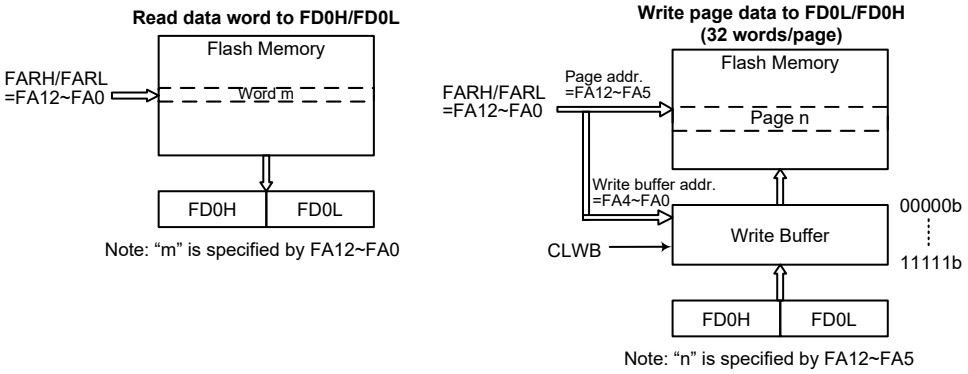
Erase Page	FARH	FARL [7:5]	FARL [4:0]
0	0000 0000	000	x xxxx
1	0000 0000	001	x xxxx
2	0000 0000	010	x xxxx
3	0000 0000	011	x xxxx
4	0000 0000	100	x xxxx
:	:	:	:
254	0001 1111	110	x xxxx
255	0001 1111	111	x xxxx

"x": Don't care

Erase Page Number and Selection – HT68FB571



Flash Memory IAP Read/Write Structure – HT68FB541



Flash Memory IAP Read/Write Structure – HT68FB571

Write Buffer

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

The write buffer size is 32 words corresponding to a page respectively. The write buffer address is mapped to a specific flash memory page specified by the memory address bits, FA11~FA5 or FA12~FA5. The data written into the FD0L and FD0H registers will be loaded into the write buffer. When data is written into the high byte data register, FD0H, it will result in the data stored in the high and low byte data registers both being written into the write buffer. It will also cause the flash memory address to be increased by one, after which the new address will be loaded into the FARH and FARL address registers. When the flash memory address reaches the page boundary, 11111b of a page with 32 words, the address will now not be increased but will stop at the last address of the page. At this point a new page address should be specified for any other erase/write operations.

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

IAP Flash Program Memory Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
FCR	CFWEN	FMOD2	FMOD1	FMOD0	BWT	FWT	FRDEN	FRD
FRCR	D7	D6	—	D4	—	—	—	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH (HT68FB541)	—	—	—	—	FA11	FA10	FA9	FA8
FARH (HT68FB571)	—	—	—	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Register List

- **FARL Register**

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

Bit 7~0 **FA7~FA0**: Flash memory address bit 7 ~ bit 0

- **FARH Register – HT68FB541**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **FA11~FA8**: Flash memory address bit 11 ~ bit 8

- **FARH Register – HT68FB571**

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

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **FA12~FA8**: Flash memory address bit 12 ~ bit 8

• **FD0L Register**

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

Bit 7~0 **D7~D0**: The first Flash memory data word bit 7 ~ bit 0

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

• **FD0H Register**

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

Bit 7~0 **D15~D8**: The first Flash memory data word bit 15 ~ bit 8

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

• **FD1L Register**

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

Bit 7~0 **D7~D0**: The second Flash memory data word bit 7 ~ bit 0

• **FD1H Register**

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

Bit 7~0 **D15~D8**: The second Flash memory data word bit 15 ~ bit 8

• **FD2L Register**

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

Bit 7~0 **D7~D0**: The third Flash memory data word bit 7 ~ bit 0

• **FD2H Register**

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

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

• **FD3L Register**

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

Bit 7~0 **D7~D0**: The fourth Flash memory data word bit 7 ~ bit 0

• **FD3H Register**

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

Bit 7~0 **D15~D8**: The fourth Flash memory data word bit 15 ~ bit 8

• **FCR Register**

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

Bit 7 **CFWEN**: Flash memory erase/write function enable control
0: Flash memory erase/write function is disabled
1: Flash memory erase/write function has been successfully enabled

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

Bit 6~4 **FMOD2~FMOD0**: Flash memory mode selection
000: Write mode
001: Page erase mode
010: Reserved
011: Read Mode
100: Reserved
101: Reserved
110: Flash memory erase/write function enable mode
111: Reserved

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

Bit 3 **BWT**: Flash memory erase/write function enable procedure trigger
0: Erase/write function enable procedure is not triggered or procedure timer times out
1: Erase/write function enable procedure is triggered and procedure timer starts to count

This bit is used to activate the flash memory erase/write function enable procedure and an internal timer. It is set by the application programs and cleared to 0 by the hardware after 1ms when the internal timer times out. The correct patterns must be written into the FD1L/FD1H, FD2L/FD2H and FD3L/FD3H register pairs respectively as soon as possible after the BWT bit is set high.

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

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

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

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

• **FRCR Register**

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

Bit 7 **D7**: Reserved bit, cannot be used and must be fixed at "0"

Bit 6 **D6**: Reserved bit, cannot be used

Bit 5 Unimplemented, read as "0"

Bit 4 **D4**: Reserved bit, cannot be used and must be fixed at "0"

Bit 3~1 Unimplemented, read as "0"

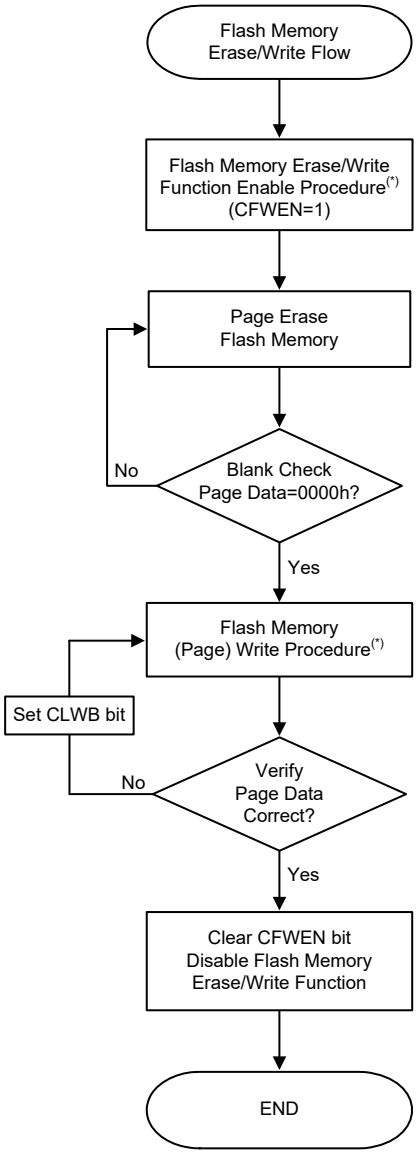
Bit 0 **CLWB**: Flash memory write buffer clear control
0: Do not initiate a Write Buffer Clear process or indicating that a Write Buffer Clear process has completed
1: Initiate Write Buffer Clear process
This bit is set by software and cleared by hardware when the Write Buffer Clear process has completed.

Flash Memory Erase/Write Flow

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

Flash Memory Erase/Write Flow Descriptions:

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



Flash Memory Erase/Write Flow

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

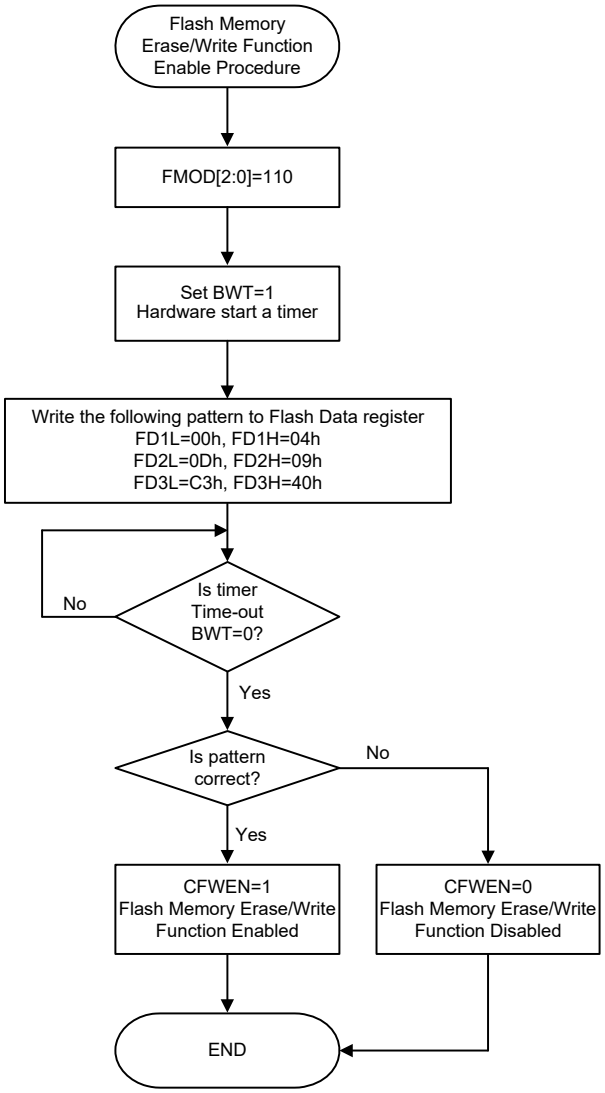
Flash Memory Erase/Write Function Enable Procedure

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

Flash Memory Erase/Write Function Enable Procedure Description

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

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



Flash Memory Erase/Write Function Enable Procedure

Flash Memory Write Procedure

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

The write buffer size is 32 words, known as a page, whose address is mapped to a specific flash memory page specified by the memory address bits, FA11~FA5 or FA12~FA5. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits, FA11~FA5 or FA12~FA5, specify.

Flash Memory Consecutive Write Description

The maximum amount of write data is 32 words for each write operation. The write buffer address will be automatically increased by one when consecutive write operations are executed. The start address of a specific page should first be written into the FARL and FARH registers. Then the data word should first be written into the FD0L register and then the FD0H register. At the same time the write buffer address will be increased by one and then the next data word can be written into the FD0L and FD0H registers for the next address without modifying the address register pair, FARH and FARL. When the write buffer address reaches the page boundary the address will not be further incremented but will stop at the last address of the page.

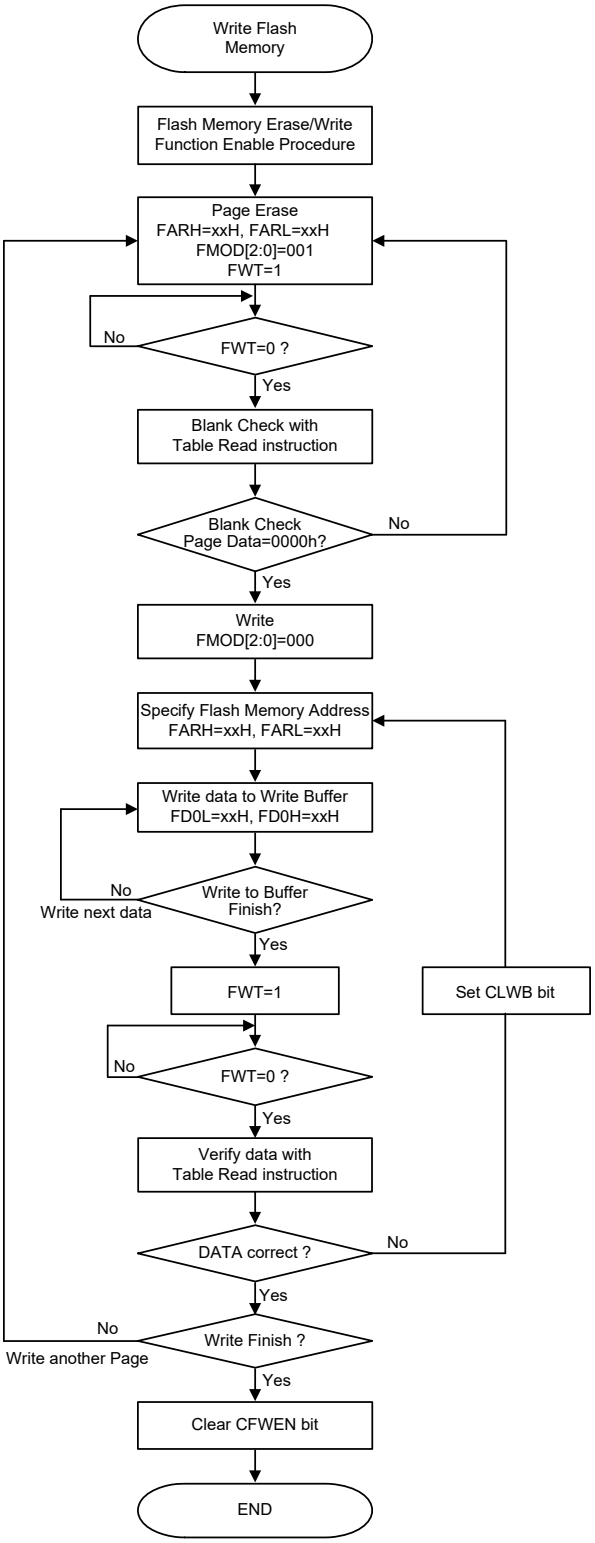
1. Activate the “Flash Memory Erase/Write function enable procedure”. Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the “Flash Memory Erase/Write function enable procedure” for more details.
2. Set the FMOD field to “001” to select the erase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

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

   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to “000” to select the write operation.
5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L and FD0H registers within a page as specified by their consecutive addresses. The maximum written data number is 32 words.
6. Set the FWT bit high to write the data words from the write buffer to the flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

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

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



Flash Memory Consecutive Write Procedure

Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.
2. It will take a typical time of 2.2ms for the FWT bit state changing from high to low.

Flash Memory Non-Consecutive Write Description

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

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

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

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

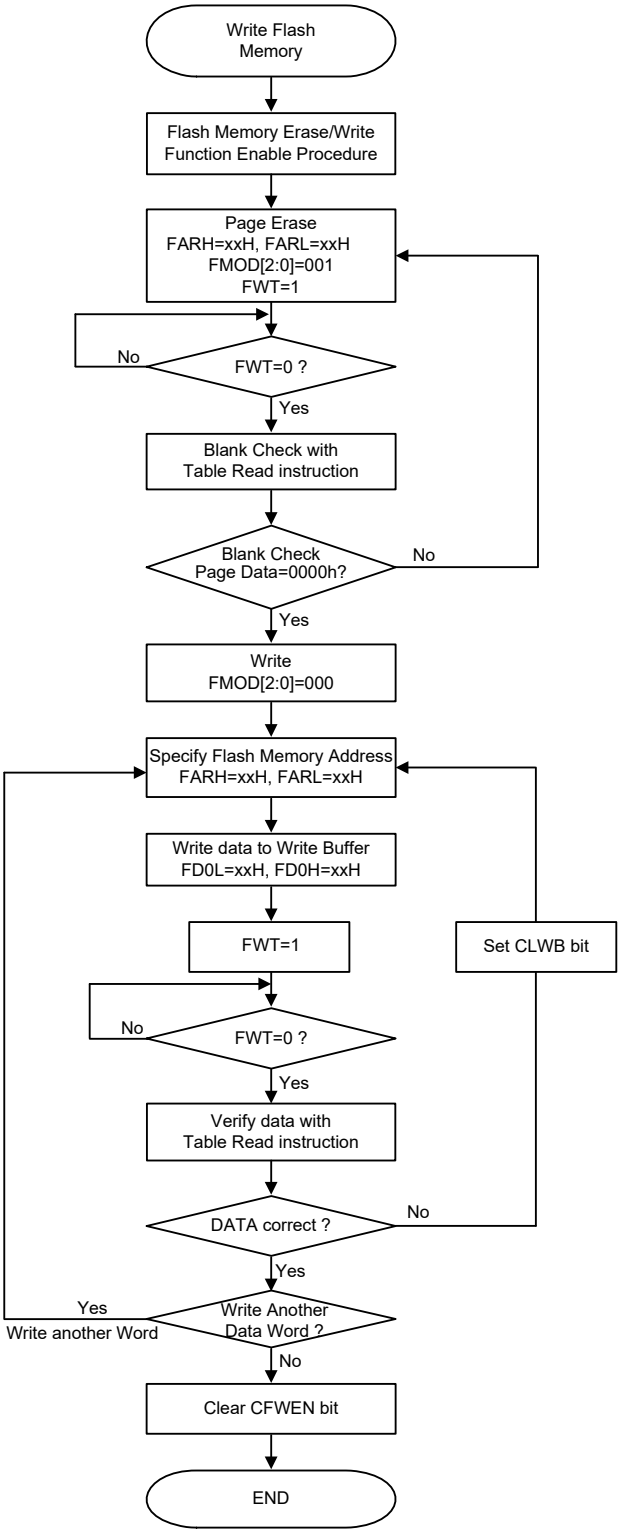
   Go to step 4 if the erase operation is successful.
4. Set the FMOD field to "000" to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
6. Set the FWT bit high to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

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

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

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

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



Flash Memory Non-Consecutive Write Procedure

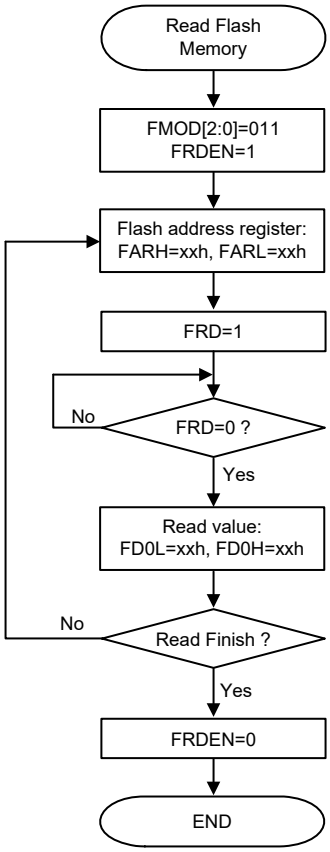
Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.
2. It will take a typical time of 2.2ms for the FWT bit state changing from high to low.

Important Points to Note for Flash Memory Write Operations

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

Flash Memory Read Procedure

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



Flash Memory Read Procedure

Note: 1. When the read operation is successfully activated, all CPU operations will temporarily cease.

2. It will take a typical time of three instruction cycles for the FRD bit state changing from high to low.

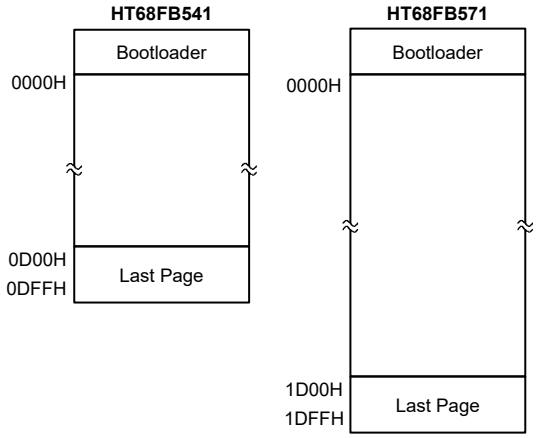
In System Programming – ISP

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

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

ISP Bootloader

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



Flash Program Memory Structure including Bootloader

Data Memory

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

Categorized into three types, the first of these is an area of RAM, known as the Special Function Data Memory. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is 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 third area is reserved for the independent LED Data Memory which is only available for the HT68FB571 device.

Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to the correct value if using the indirect addressing method.

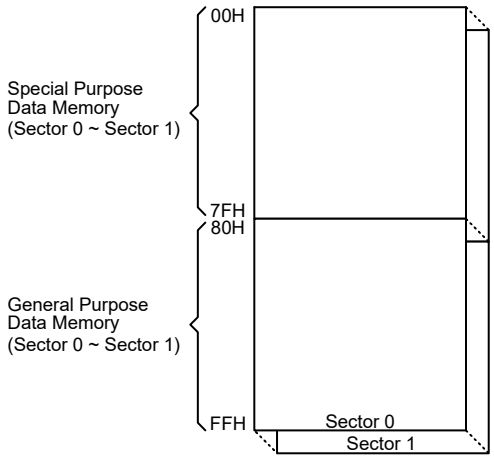
Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide RAM. Each of the Data Memory Sector is categorized into two types, the special Purpose Data Memory and the General Purpose Data Memory.

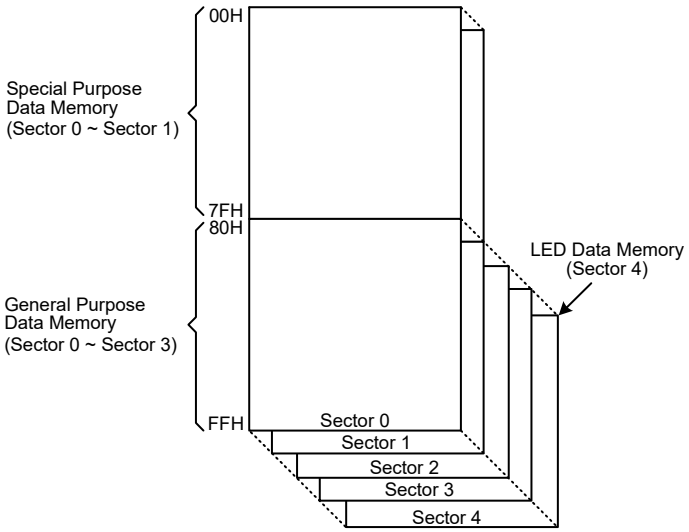
The address range of the Special Purpose Data Memory for the devices is from 00H to 7FH. The General Purpose Data Memory for the devices and the independent LED Data Memory for the HT68FB571 device address range are from 80H to FFH.

Device	Special Purpose Data Memory	General Purpose Data Memory		LED Data Memory	
	Located Sectors	Capacity	Sector: Address	Capacity	Sector: Address
HT68FB541	0, 1	256×8	0: 80H~FFH 1: 80H~FFH	—	—
HT68FB571	0, 1	512×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH	128×8	4: 80H~FFH

Data Memory Summary



Data Memory Structure – HT68FB541



Data Memory Structure – HT68FB571

Data Memory Addressing

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

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

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH		
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	INTC0	
11H	INTC1	INTEG
12H	INTC2	
13H	INTC3	
14H	PA	PAS0
15H	PAC	PAS1
16H	PAPU	
17H	PAWU	
18H	PB	PBS0
19H	PBC	PBS1
1AH	PBPU	
1BH		
1CH	PC	PCS0
1DH	PCC	
1EH	PCPU	
1FH	PCWU	
20H		
21H		
22H		
23H		
24H	PF	PFS0
25H	PFC	
26H	PFPU	
27H		
28H	LVRC	
29H	LVDC	
2AH	PMPS	
2BH	RSTC	
2CH		
2DH		
2EH	SLEDC0	
2FH	SLEDC1	
30H	TMR0C	
31H	TMR0L	
32H	TMR0H	
33H	TMR1C	
34H	TMR1L	
35H	TMR1H	
36H	WDTC	
37H	SPIC0	
38H	SPIC1	
39H	SPID	
3AH	CMP0C	
3BH	CMP1C	
3CH		
3DH		
3EH	EEA	EEC
3FH	EED	

	Sector 0	Sector 1
40H		FRCR
41H		FCR
42H	FARL	FARH
43H	FD0L	FD0H
44H	FD1L	FD1H
45H	FD2L	FD2H
46H	FD3L	FD3H
47H		
48H		
49H		
4AH		
4BH		
4CH	KEYDATA0	
4DH	KEYDATA1	
4EH	KEYDATA2	
4FH	KEYDATA3	PWMCTL0
50H	KEYDATA4	PWMCTL1
51H	KEYDATA5	PWMCTL2
52H	KEYDATA6	PWMCTL3
53H	KEYDATA7	
54H		LMCE
55H		
56H		
57H		
58H		LMCAR
59H		LMCBR
5AH		LMCCR
5BH		PWMC0A
5CH		PWMC0B
5DH		PWMC0C
5EH		PWMC1A
5FH		PWMC1B
60H	SCC	PWMC1C
61H	HIRCC	PWMC2A
62H	PLLC	PWMC2B
63H	SYSC	PWMC2C
64H	USB_STAT	PWMC3A
65H	UINT	PWMC3B
66H	USC	PWMC3C
67H	UESR	PWMC4A
68H	UCC	PWMC4B
69H	AWR	PWMC4C
6AH	STL	PWMC5A
6BH	SIES	PWMC5B
6CH	MISC	PWMC5C
6DH	SETIO	PWMC6A
6EH	FRNUM0	PWMC6B
6FH	FRNUM1	PWMC6C
70H	FIFO0	PWMC7A
71H	FIFO1	PWMC7B
72H	FIFO2	PWMC7C
73H	FIFO3	
74H		
75H	TB0C	
76H	TB1C	
77H	PSCR	
78H		
79H		
7AH		
7BH		
7CH		
7DH		
7EH		
7FH		

: Unused, read as 00H

Special Purpose Data Memory – HT68FB541

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH		
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	INTC0	
11H	INTC1	INTEG
12H	INTC2	
13H	INTC3	
14H	PA	PAS0
15H	PAC	PAS1
16H	PAPU	
17H	PB	PBS0
18H	PBC	PBS1
19H	PBPU	
1AH	PC	PCS0
1BH	PCC	
1CH	PCPU	
1DH	PD	PDS0
1EH	PDC	PDS1
1FH	PDPU	INLVC
20H	PDWU	
21H	PE	PES0
22H	PEC	PES1
23H	PEPU	
24H	PF	PFS0
25H	PFC	PFS1
26H	PFPU	
27H		
28H	LVRC	
29H	LVDC	
2AH	PMPS	
2BH	RSTC	
2CH		
2DH		
2EH	SLEDC	
2FH		
30H	TMR0C	
31H	TMR0L	
32H	TMR0H	
33H	TMR1C	
34H	TMR1L	
35H	TMR1H	
36H	WDTC	
37H	SPIC0	
38H	SPIC1	
39H	SPID	
3AH		
3BH		
3CH		
3DH		
3EH	EEA	EEC
3FH	EED	

	Sector 0	Sector 1
40H		FRCR
41H		FCR
42H	FARL	FARH
43H	FD0L	FD0H
44H	FD1L	FD1H
45H	FD2L	FD2H
46H	FD3L	FD3H
47H		
48H		
49H		
4AH		
4BH		
4CH	KEYDATA0	
4DH	KEYDATA1	
4EH	KEYDATA2	
4FH	KEYDATA3	PWMCTL0
50H	KEYDATA4	PWMCTL1
51H	KEYDATA5	PWMCTL2
52H	KEYDATA6	PWMCTL3
53H	KEYDATA7	PWMCTL4
54H	KEYDATA8	LM0CE
55H	KEYDATA9	LM1CE
56H	KEYDATA10	LM2CE
57H	KEYDATA11	LM3CE
58H	KEYDATA12	LM0CAR
59H	KEYDATA13	LM0CBR
5AH	KEYDATA14	LM1CAR
5BH	KEYDATA15	LM1CBR
5CH	KEYDATA16	LM2CAR
5DH	KEYDATA17	LM2CBR
5EH	KEYDATA18	LM3CAR
5FH	KEYDATA19	LM3CBR
60H	SCC	
61H	HIRCC	
62H	PLLC	
63H	SYSC	
64H	USB_STAT	
65H	UINT	
66H	USC	
67H	UESR	
68H	UCC	
69H	AWR	
6AH	STL	
6BH	SIES	
6CH	MISC	
6DH	SETIO	
6EH	FRNUM0	
6FH	FRNUM1	
70H	FIFO0	
71H	FIFO1	
72H	FIFO2	
73H	FIFO3	
74H		
75H	TB0C	
76H	TB1C	
77H	PSCR	
78H		
79H		
7AH		
7BH		
7CH		
7DH		
7EH		
7FH		

: Unused, read as 00H

Special Purpose Data Memory – HT68FB571

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, IAR2

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

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

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

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

Indirect Addressing Program Example 1

```
data .section ´data´
adres1   db ?
adres2   db ?
adres3   db ?
adres4   db ?
block    db ?
code .section at 0 ´code´
org 00h
start:
     mov a, 04h              ; 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                 ; increase memory pointer
     sdz block               ; check if last memory location has been cleared
     jmp loop
continue:
```

Indirect Addressing Program Example 2

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

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

Direct Addressing Program Example using Extended Instructions

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

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

Accumulator – ACC

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

Program Counter Low Byte Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

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

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

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

- SC is the result of the "XOR" operation which is performed by the OV flag and the MSB of the current instruction operation result.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.
- 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	SC	CZ	TO	PDF	OV	Z	AC	C
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
POR	x	x	0	0	x	x	x	x

"x": Unknown

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

Bit 6 **CZ**: The operational result of different flags for different instructions
For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.
For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation zero flag.
For other instructions, the CZ flag will not be affected.

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

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

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

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

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

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

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 64×8 bits for the devices. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and a data register in Sector 0 and a single control register in Sector 1.

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Sector 0, they can be directly accessed in the same way as many other Special Function Registers. The EEC register however, being located in Sector 1, can only be read from or written to indirectly using the MP1L/MP1H or MP2L/MP2H Memory Pointer and Indirect Addressing Register, IAR1/IAR2. As the EEC control register is located at address 3EH in Sector 1, the MP1L or MP2L Memory Pointer must first be set to the value 3EH and the MP1H or MP2H Memory Pointer high byte set to the value, 01H, before any operations on the EEC register are executed.

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

EEPROM Register List

- **EEA Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **EEA5~EEA0**: Data EEPROM address bit 5 ~ bit 0

- **EED Register**

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

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

• **EEC Register**

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

Bit 7~4 Unimplemented, read as "0"

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

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

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

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

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

Reading Data from the EEPROM

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

Writing Data to the EEPROM

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

Write Protection

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

EEPROM Interrupt

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

Programming Considerations

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

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

Programming Examples

Reading data from the EEPROM – polling method

```
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
MOV A, 03EH               ; setup memory pointer MP1L
MOV MP1L, A               ; MP1L points to EEC register
MOV A, 01H                ; setup memory pointer MP1H
MOV MP1H, A
SET IAR1.1                ; set RDEN bit, enable read operations
SET IAR1.0                ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0                ; check for read cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read if no more read operations are required
CLR MP1H
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

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

Writing Data to the EEPROM – Polling Method

```
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
MOV A, 03EH               ; setup memory pointer MP1L
MOV MP1L, A               ; MP1L points to EEC register
MOV A, 01H                ; setup memory pointer MP1H
MOV MP1H, A
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit - executed immediately
                          ; after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2                ; check for write cycle end
JMP BACK
CLR MP1H
```

Oscillators

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

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options are selected through the relevant control registers. The higher frequency oscillator 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 devices have the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

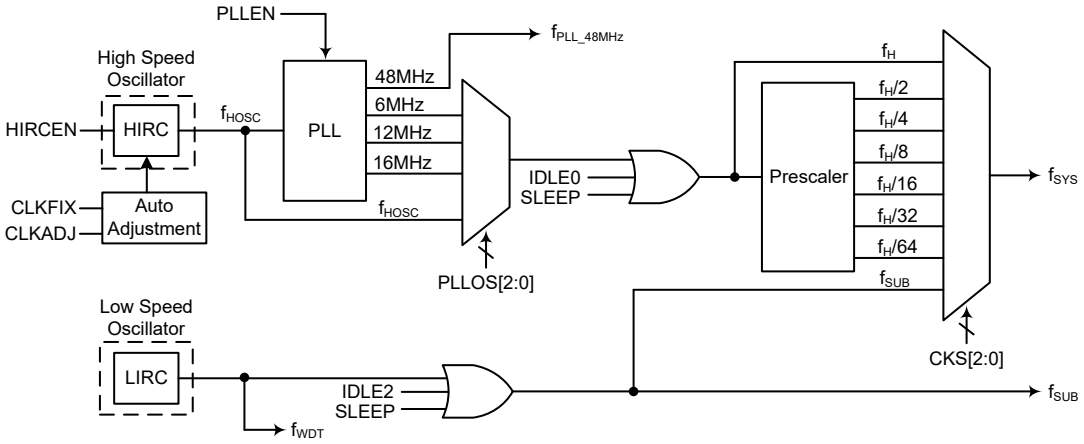
Type	Name	Frequency
Internal High Speed RC	HIRC	12MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two oscillator sources, one high speed oscillator and one low speed oscillator. The high speed system clock is the internal 12MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

The frequency of the slow speed or high speed system clock is also determined using the CKS2~CKS0 bits in the SCC register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. In addition, the internal PLL frequency generator can be enabled by a software control bit to generate various frequencies for the USB interface and system clock.



System Clock Configurations

Internal PLL Frequency Generator

The internal PLL frequency generator is used to generate the frequency for the USB interface and the system clock. This PLL generator can be enabled or disabled by the PLL control bit PLLEN in the PLLC register. After a power on reset, the PLL control bit will be set to 1 to turn on the PLL generator. The PLL generator will provide the fixed 48MHz frequency for the USB operating frequency and another frequency for the system clock source which can be 6MHz, 12MHz or 16MHz. The selection of this system frequency is implemented using the PLLEN and PLLOS2~PLLOS0 bits in the PLLC register.

The following table illustrates the high frequency system clock f_H selected by the related control bits.

PLLEN	PLLOS2	PLLOS1	PLLOS0	f_H
0	x	x	x	$f_{HOSC}=f_{HIRC}$
1	0	x	x	$f_{HOSC}=f_{HIRC}$
1	1	0	0	f_{PLL}=6MHz
1	1	0	1	f_{PLL}=12MHz
1	1	1	0	f_{PLL}=16MHz
1	1	1	1	Reserved

"x": Don't care

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 12MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Internal 32kHz Oscillator – LIRC

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

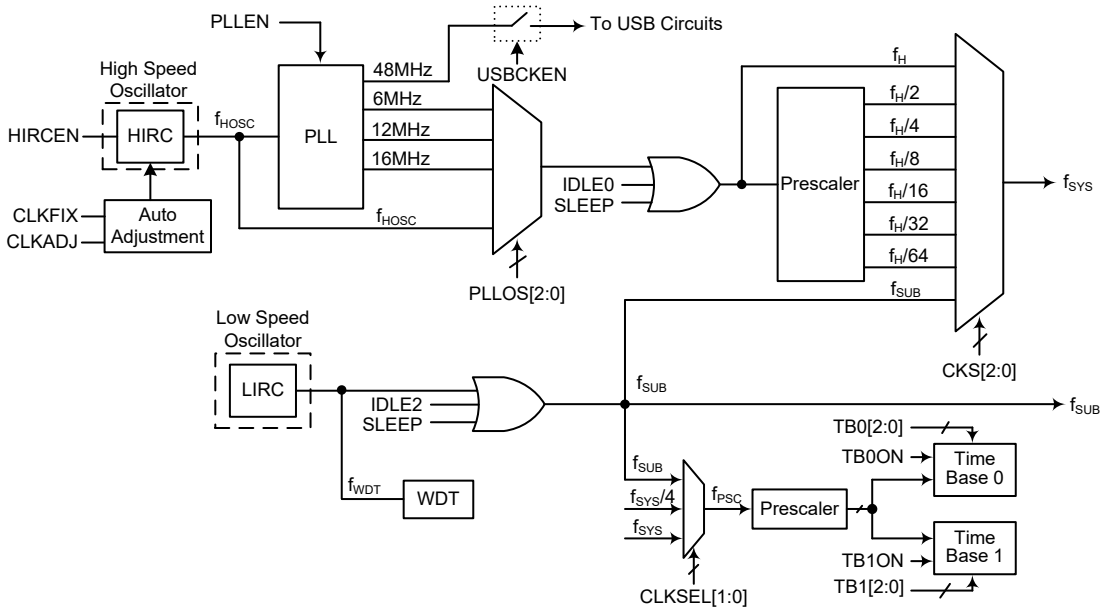
Operating Modes and System Clocks

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

System Clocks

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

The main system clock, can come from a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock can be sourced from an HIRC oscillator or the PLL output clock, selected via configuring the PLLEN and PLLOS2~PLLOS0 bits in the PLLC register. 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 $f_H/2$~$f_H/64$.



Device Clock Configurations

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

System Operation Modes

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

Operation Mode	CPU	Register Setting			f_{SYS}	f_H	f_{SUB}	f_{WDT}	f_{PLL}
		FHIDEN	FSIDEN	CKS2~CKS0					
NORMAL	On	x	x	000~110	f_H~f_H/64	On	On	On	On
SLOW	On	x	x	111	f_{SUB}	On/Off (1)	On	On	On/Off (3)
IDLE0	Off	0	1	000~110	Off	Off	On	On	Off
				111	On				
IDLE1	Off	1	1	xxx	On	On	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off	On/Off (2)	On
				111	Off				
SLEEP	Off	0	0	xxx	Off	Off	Off	On/Off (2)	Off

"x": Don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.
2. The f_{WDT} clock can be switched on or off which is controlled by the WDT function being enabled or disabled.
3. The f_{PLL} clock can be switched on or off which is controlled by the PLLEN bit in the PLLC register in the SLOW mode.
4. The USB function is inactive in the IDLE0 and SLEEP mode.

NORMAL Mode

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

SLOW Mode

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

SLEEP Mode

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

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
HIRCC	CLKADJ	CLKADJF	CLKFIX	—	—	—	HIRCF	HIRCEN
PLLC	D7	—	—	PLLOS2	PLLOS1	PLLOS0	PLLF	PLLEN

System Operating Mode Control Register List

- **SCC Register**

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

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

Bit 4~2 Unimplemented, read as "0"

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

Bit 0 **FSIDEN**: Low frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing a "HALT" instruction.

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

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

• HIRCC Register

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

Bit 7 **CLKADJ**: HIRC clock automatic adjustment function control in the USB mode
0: Disable
1: Enable
Note that if users select the HIRC as the system clock, the CLKADJ bit must be set to 1 to adjust the PLL frequency automatically.

Bit 6 **CLKADJF**: HIRC clock automatic adjustment stable flag
0: Unstable
1: Stable
The CLKADJF bit indicates whether the HIRC frequency adjusting operation is completed or not when the CLKADJ bit is set to 1. Users can continuously monitor the CLKADJF bit by application programs to make sure that the HIRC frequency accuracy is stably adjusted in the range of ±0.25%.
The CLKADJF bit can be cleared by the application program.

Bit 5 **CLKFIX**: HIRC clock fix automatic adjustment function control
0: Disable
1: Enable
Note that when CLKADJF=1, the CLKFIX bit can be set high to fix the HIRC frequency accuracy in the range of ±0.25%.

Bit 4~2 Unimplemented, read as "0"

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

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

• PLLC Register

Bit	7	6	5	4	3	2	1	0
Name	D7	—	—	PLLOS2	PLLOS1	PLLOS0	PLLF	PLLEN
R/W	R/W	—	—	R/W	R/W	R/W	R	R/W
POR	0	—	—	0	0	0	0	1

Bit 7 **D7**: Reserved bit, cannot be used and must be fixed at "0"

Bit 6~5 Unimplemented, read as "0"

Bit 4~2 **PLLOS2~PLLOS0**: f_{HOSC} or f_{PLL} for f_H clock source selection
0xx: f_{HOSC}
100: f_{PLL}=6MHz
101: f_{PLL}=12MHz
110: f_{PLL}=16MHz
111: Reserved

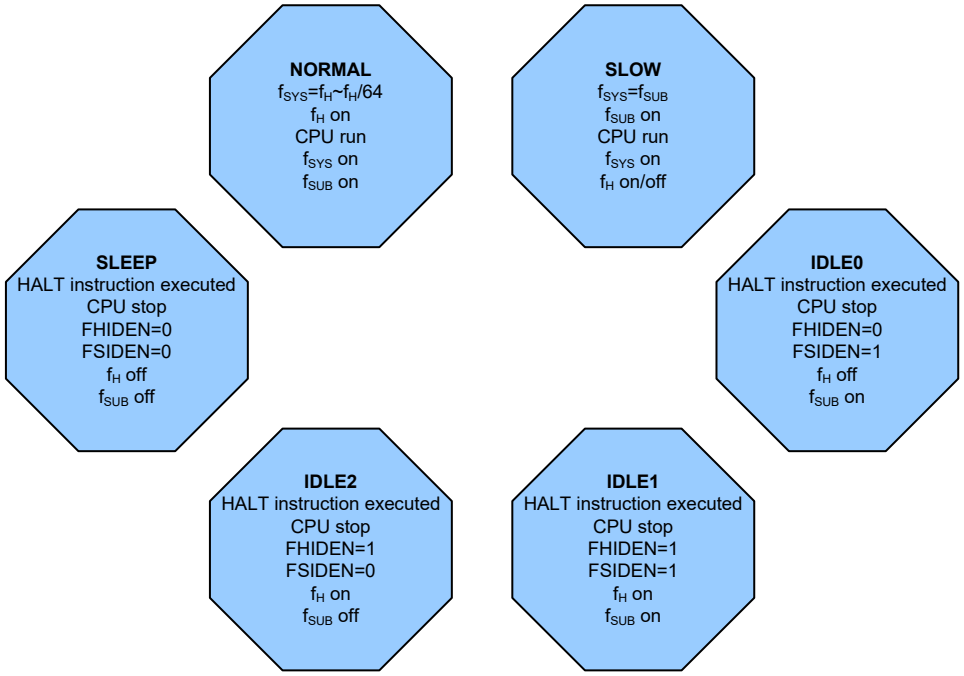
Bit 1 **PLLF**: PLL clock stable flag
0: Unstable
1: Stable
This bit is used to indicate whether the PLL clock is stable or not. When the PLLEN bit is set to 1 to enable the PLL clock, the PLLF bit will first be cleared to 0 and then set to 1 after the PLL clock is stable.

Bit 0 **PLLEN**: PLL enable control
0: Disable
1: Enable

Operating Mode Switching

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

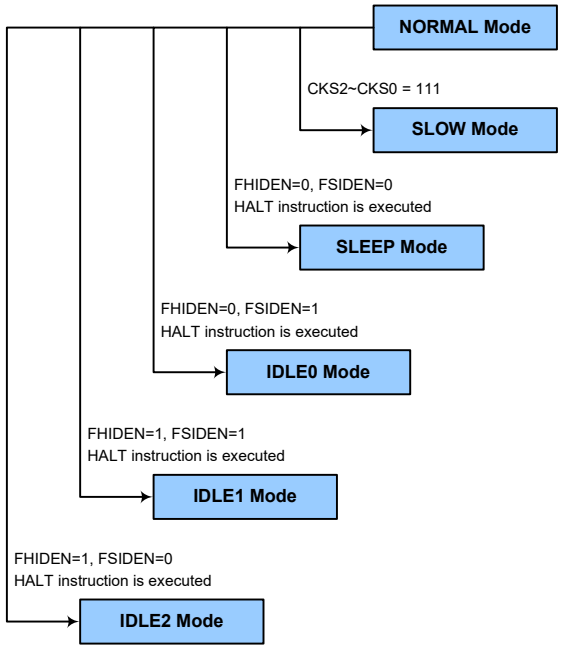
In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enter the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



NORMAL Mode to SLOW Mode Switching

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

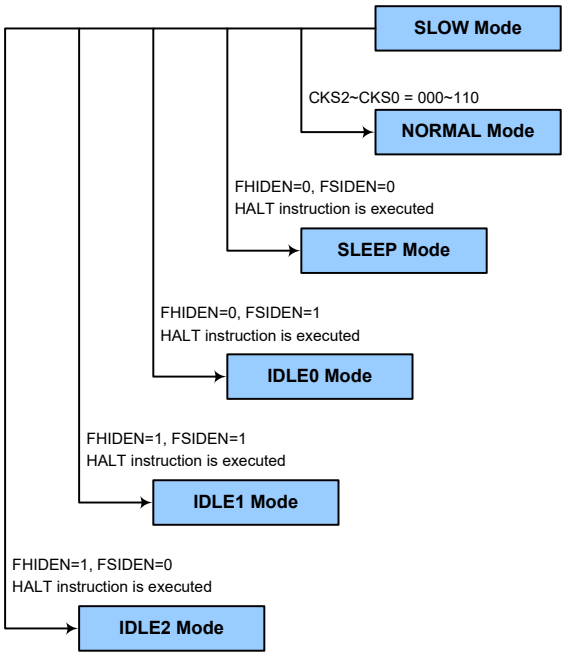
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to NORMAL Mode Switching

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

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



Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

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

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

Entering the IDLE2 Mode

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

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

Standby Current Considerations

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

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

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

Wake-up

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

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

- An external pin reset
- A USB reset signal reset
- An external falling edge on I/O port with wake-up ability
- A system interrupt
- A WDT overflow

If the system is woken up by an external pin 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 Port A and the PC0 pin of the HT68FB541 and each pin on Port D of the HT68FB571 can be setup using the PxWU register to permit a negative transition on the pin to wake-up the system. When a pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

Watchdog Timer

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

Watchdog Timer Clock Source

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

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period, the enable/disable operation as well as MCU 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
Other values: Reset MCU
When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set high.

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

• **RSTFC Register**

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

"x": Unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Refer to the $\overline{RES}$ Pin Reset section

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

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

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

Watchdog Timer Operation

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

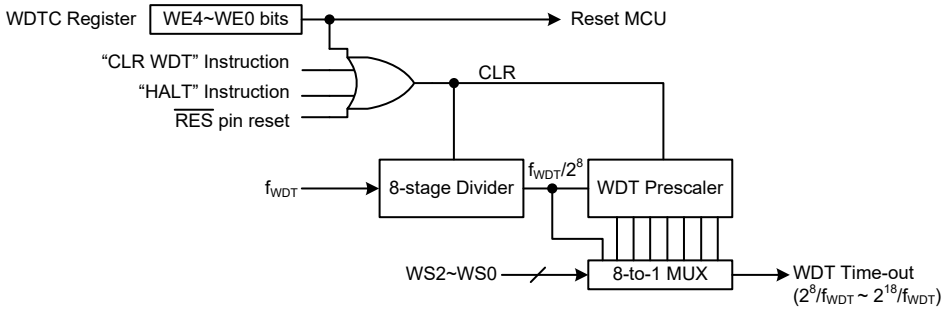
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other values	Reset MCU

Watchdog Timer Function 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 and PDF bits in the status register will be set and only the Program Counter and Stack Pointer will be reset. Four methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bits, the second is using the Watchdog Timer software clear instruction, the third is via a HALT instruction and the fourth is an external hardware reset, which means a low level on the external $\overline{RES}$ pin.

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

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



Watchdog Timer

Reset and Initialisation

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

In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the device is running. One example of this is where after power has been applied and the device 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 device to proceed with normal operation after the reset line is allowed to return high.

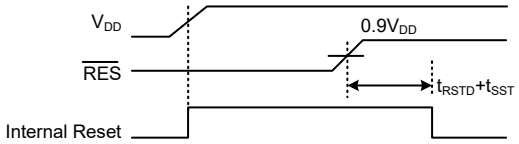
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{RES}$ reset is implemented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

Reset Functions

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

Power-on Reset

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



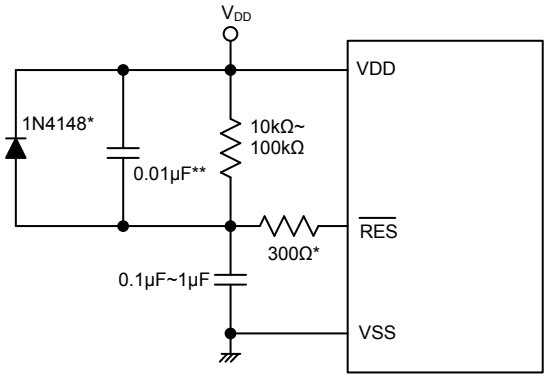
Power-on Reset Timing Chart

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

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

For most applications a resistor connected between VDD and the $\overline{\text{RES}}$ pin and a capacitor connected between VSS and the $\overline{\text{RES}}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{\text{RES}}$ pin should be kept as short as possible to minimise any stray noise interference.

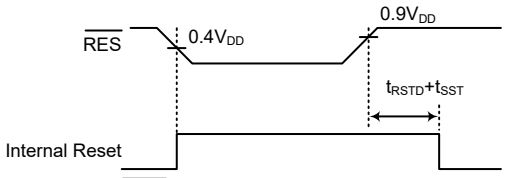
For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



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

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

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.



RES Reset Timing Chart

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

RSTC7~RSTC0 Bits	Reset Function
01010101B	I/O pin
10101010B	RES
Any other value	Reset MCU

Internal Reset Function Control

- **RSTC Register**

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

Bit 7~0 **RSTC7~RSTC0**: Reset function control
01010101: I/O pin
10101010: RES pin
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET}, and the RSTF bit in the RSTFC register will be set to 1.
All resets will reset this register to POR value except the WDT time out hardware warm reset.

- **RSTFC Register**

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

"x": Unknown

Bit 7~4 Unimplemented, read as "0"

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

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

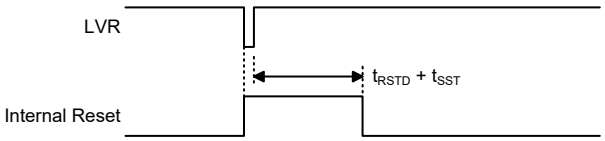
Bit 1 **LRF**: LVR control register software reset flag
Refer to the Low Voltage Reset section

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

Low Voltage Reset – LVR

Each of the microcontrollers contain a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level.

The LVR function is always enabled in the NORMAL and SLOW mode 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 in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVR/LVD Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set 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 IDLE/SLEEP mode.



Low Voltage Reset Timing Chart

• LVRC Register

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

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

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

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

• **RSTFC Register**

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

"x": Unknown

Bit 7~2 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Refer to the $\overline{\text{RES}}$ Pin Reset section

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.

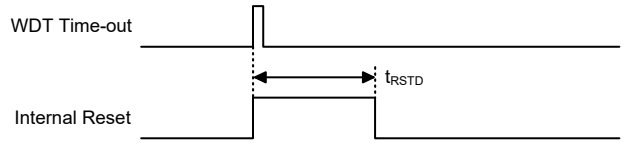
Bit 0 **WRF**: WDT control register software reset flag
Refer to the Watchdog Timer section

USB Reset

The devices contain a USB circuit, a reset signal can be generated by properly configuring the USB control register to reset the whole device. Refer to the "USB Interface" section for more associated details.

Watchdog Time-out Reset during Normal Operation

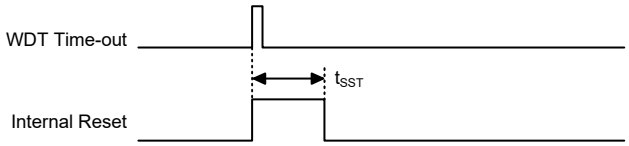
When the Watchdog time-out Reset during normal operations in the NORMAL or SLOW mode occurs, the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

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

"u": Unchanged

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

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Bases	Cleared after reset, WDT begins counting
Timer/Event Counters	Timer/Event Counters 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.

HT68FB541

Register	Power-on Reset	$\overline{RES}$ Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)	USB Reset (Normal Operation)	USB Reset (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
IAR1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---- xxxx	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
STATUS	xx00 xxxx	uuuu uuuu	uuuu uuuu	uu1u uuuu	uu11 uuuu	xxuu uuuu	xxuu uuuu
IAR2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP2L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP2H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
RSTFC	---- 0x00	---- uuuu	---- u1uu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000
INTC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
INTC2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
INTC3	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
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

Register	Power-on Reset	RES Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)	USB Reset (Normal Operation)	USB Reset (IDLE/SLEEP)
PAPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PAWU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PB	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu	--11 1111	--11 1111
PBC	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu	--11 1111	--11 1111
PBPU	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
PC	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u	---- ---1	---- ---1
PCC	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u	---- ---1	---- ---1
PCPU	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
PCWU	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
PF	---- -111	---- -111	---- -111	---- -111	---- -uuu	---- -111	---- -111
PFC	---- -111	---- -111	---- -111	---- -111	---- -uuu	---- -111	---- -111
PFPU	---- -000	---- -000	---- -000	---- -000	---- -uuu	---- -000	---- -000
LVRC	0101 0101	0101 0101	uuuu uuuu	0101 0101	uuuu uuuu	0101 0101	0101 0101
LVDC	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
PMPS	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu	---- 0000	---- 0000
RSTC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu	uuuu uuuu	uuuu uuuu
SLEDC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
SLEDC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TMR0C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu	0000 1000	0000 1000
TMR0L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TMR0H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TMR1C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu	0000 1000	0000 1000
TMR1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TMR1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
WDTC	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu	0101 0011	0101 0011
SPIC0	111- --00	111- --00	111- --00	111- --00	uuu- --uu	111- --00	111- --00
SPIC1	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
SPID	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
CMP0C	-000 ---1	-000 ---1	-000 ---1	-000 ---1	-uuu ---u	-000 ---1	-000 ---1
CMP1C	-000 ---1	-000 ---1	-000 ---1	-000 ---1	-uuu ---u	-000 ---1	-000 ---1
EEA	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
EED	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FARL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD0L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD2L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD3L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA0	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
KEYDATA1	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
KEYDATA2	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
KEYDATA3	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
KEYDATA4	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
KEYDATA5	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
KEYDATA6	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
KEYDATA7	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
SCC	010- --00	010- --00	010- --00	010- --00	uuu- --uu	010- --00	010- --00
HIRCC	000- --01	uuu- --01	uuu- --01	uuu- --01	uuu- --uu	uuu- --01	uuu- --01
PLLC	0--0 0001	0--u uu0u	0--u uu0u	0--u uu0u	u--u uuuu	0--u uu0u	0--u uu0u
SYSC	0001 --0x	0001 --0x	0001 --0x	0001 --0x	uuuu --ux	0001 --0x	0001 --0x
USB_STAT	11xx 0001	uuxx uuuu	uuxx uuuu	uuxx uuuu	uuxx uuuu	uuxx uuuu	uuxx uuuu
UINT	---- 0000	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
USC	1000 xxxx	uuuu xuux	uuuu xuux	uuuu xuux	uuuu xuux	uuuu 0100	uuuu 0100
UESR	---- xxxx	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
UCC	0-0x 0-xx	u-uu u-uu	u-uu u-uu	u-uu u-uu	u-uu u-uu	u-uu u-00	u-uu u-00

Register	Power-on Reset	RES Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)	USB Reset (Normal Operation)	USB Reset (IDLE/SLEEP)
AWR	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
STL	---- xxxx	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- 0000	---- 0000
SIES	xxxx xxxx	uxxx xuuu	uxxx xuuu	uxxx xuuu	uxxx xuuu	0000 0000	0000 0000
MISC	xxx0 -xxx	xxuu -uxx	xxuu -uxx	xxuu -uxx	xxuu -uxx	000u -000	000u -000
SETIO	---- 1110	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
FRNUM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
FRNUM1	---- -xxx	---- -xxx	---- -xxx	---- -xxx	---- -uuu	---- -xxx	---- -xxx
FIFO0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO3	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TB0C	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu	0--- -000	0--- -000
TB1C	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu	0--- -000	0--- -000
PSCR	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
INTEG	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu	---- 0000	---- 0000
PAS0	00-- 00--	00-- 00--	00-- 00--	00-- 00--	uu-- uu--	00-- 00--	00-- 00--
PAS1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PBS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PBS1	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu	---- 0000	---- 0000
PCS0	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
PFS0	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
EEC	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu	---- 0000	---- 0000
FRCR	00-0 ---0	00-u ---0	00-u ---0	00-u ---0	uu-u ---u	00-u ---0	00-u ---0
FCR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FARH	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu	---- 0000	---- 0000
FD0H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD2H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD3H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMCTL0	0000 00-0	0000 00-0	0000 00-0	0000 00-0	uuuu uu-u	0000 00-0	0000 00-0
PWMCTL1	0000 00-0	0000 00-0	0000 00-0	0000 00-0	uuuu uu-u	0000 00-0	0000 00-0
PWMCTL2	0000 --00	0000 --00	0000 --00	0000 --00	uuuu --uu	0000 --00	0000 --00
PWMCTL3	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
LMCE	---- -000	---- -000	---- -000	---- -000	---- -uuu	---- -000	---- -000
LMCAR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
LMCBR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
LMCCR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC0A	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC0B	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC0C	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC1A	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC1B	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC1C	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC2A	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC2B	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC2C	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC3A	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC3B	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC3C	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC4A	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC4B	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC4C	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC5A	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC5B	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000

Register	Power-on Reset	RES Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)	USB Reset (Normal Operation)	USB Reset (IDLE/SLEEP)
PWMC5C	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC6A	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC6B	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC6C	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC7A	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC7B	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMC7C	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000

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

HT68FB571

Register	Power On Reset	RES Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)	USB Reset (Normal Operation)	USB Reset (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
IAR1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---x xxxx	---u uuuu	---u uuuu	---u uuuu	---u uuuu	---u uuuu	---u uuuu
STATUS	xx00 xxxx	uuuu uuuu	uuuu uuuu	uu1u uuuu	uu11 uuuu	xxuu uuuu	xxuu uuuu
IAR2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP2L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MP2H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
RSTFC	---- 0x00	---- uuuu	---- u1uu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000
INTC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
INTC2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
INTC3	--00 --00	--00 --00	--00 --00	--00 --00	--uu --uu	--00 --00	--00 --00
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PAPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PBPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PC	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u	---- ---1	---- ---1
PCC	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u	---- ---1	---- ---1
PCPU	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u	---- ---0	---- ---0
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
PDPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PDWU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PE	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PEC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PEPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PF	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PFC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PFPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000

Register	Power On Reset	RES Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)	USB Reset (Normal Operation)	USB Reset (IDLE/SLEEP)
LVRC	0101 0101	0101 0101	uuuu uuuu	0101 0101	uuuu uuuu	0101 0101	0101 0101
LVDC	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
PMPS	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu	---- 0000	---- 0000
RSTC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu	uuuu uuuu	uuuu uuuu
SLEDC	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu	---- 0000	---- 0000
TMR0C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu	0000 1000	0000 1000
TMR0L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TMR0H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TMR1C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu	0000 1000	0000 1000
TMR1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
TMR1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
WDTC	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu	0101 0011	0101 0011
SPIC0	111- --00	111- --00	111- --00	111- --00	uuu- --uu	111- --00	111- --00
SPIC1	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
SPID	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
EEA	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
EED	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FARL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD0L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD2L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD3L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA3	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA4	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA5	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA6	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA7	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA9	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA10	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA11	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA12	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA13	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA14	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA15	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA16	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA17	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA18	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
KEYDATA19	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
SCC	010- --00	010- --00	010- --00	010- --00	uuu- --uu	010- --00	010- --00
HIRCC	000- --01	uuu- --01	uuu- --01	uuu- --01	uuu- --uu	uuu- --01	uuu- --01
PLLC	0--0 0001	0--u uu0u	0--u uu0u	0--u uu0u	u--u uuuu	0--u uu0u	0--u uu0u
SYSC	0000 --0x	0000 --0x	0000 --0x	0000 --0x	uuuu --ux	0000 --0x	0000 --0x
USB_STAT	11xx 0001	uuxx uuuu	uuxx uuuu	uuxx uuuu	uuxx uuuu	uuxx uuuu	uuxx uuuu
UINT	---- 0000	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
USC	1000 xxxx	uuuu xuux	uuuu xuux	uuuu xuux	uuuu xuux	uuuu 0100	uuuu 0100
UESR	---- xxxx	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
UCC	0-0x 0-xx	u-uu u-uu	u-uu u-uu	u-uu u-uu	u-uu u-uu	u-uu u-00	u-uu u-00
AWR	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
STL	---- xxxx	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- 0000	---- 0000

Register	Power On Reset	RES Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)	USB Reset (Normal Operation)	USB Reset (IDLE/SLEEP)
SIES	xxxx xxxx	uxxx xuuu	uxxx xuuu	uxxx xuuu	uxxx xuuu	0000 0000	0000 0000
MISC	xxx0 -xxx	xxuu -uxx	xxuu -uxx	xxuu -uxx	xxuu -uxx	000u -000	000u -000
SETIO	---- 1110	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
FRNUM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
FRNUM1	---- -xxx	---- -xxx	---- -xxx	---- -xxx	---- -uuu	---- -xxx	---- -xxx
FIFO0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO3	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TB0C	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu	0--- -000	0--- -000
TB1C	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu	0--- -000	0--- -000
PSCR	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
INTEG	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu	---- 0000	---- 0000
PAS0	00-- 00--	00-- 00--	00-- 00--	00-- 00--	uu-- uu--	00-- 00--	00-- 00--
PAS1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PBS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PBS1	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
PCS0	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
PDS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PDS1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
INLVC	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
PES0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PES1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PFS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PFS1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
EEC	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu	---- 0000	---- 0000
FRCR	00-0 ---0	00-u ---0	00-u ---0	00-u ---0	uu-u ---u	00-u ---0	00-u ---0
FCR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FARH	---0 0000	---0 0000	---0 0000	---0 0000	---u uuuu	---0 0000	---0 0000
FD0H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD2H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FD3H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMCTL0	0000 00-0	0000 00-0	0000 00-0	0000 00-0	uuuu uu-u	0000 00-0	0000 00-0
PWMCTL1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMCTL2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMCTL3	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMCTL4	---- -010	---- -010	---- -010	---- -010	---- -uuu	---- -010	---- -010
LM0CE	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
LM1CE	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
LM2CE	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
LM3CE	---- --00	---- --00	---- --00	---- --00	---- --uu	---- --00	---- --00
LM0CAR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
LM0CBR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
LM1CAR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
LM1CBR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
LM2CAR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
LM2CBR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
LM3CAR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
LM3CBR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	PB5	PB4	PB3	PB2	PB1	PB0
PBC	—	—	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	—	—	—	—	—	—	PC0
PCC	—	—	—	—	—	—	—	PCC0
PCPU	—	—	—	—	—	—	—	PCPU0
PCWU	—	—	—	—	—	—	—	PCWU0
PF	—	—	—	—	—	PF2	PF1	PF0
PFC	—	—	—	—	—	PFC2	PFC1	PFC0
PFPU	—	—	—	—	—	PFPU2	PFPU1	PFPU0

"—": Unimplemented, read as "0"

I/O Logic Function Register List – HT68FB541

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	—	—	—	—	—	—	PC0
PCC	—	—	—	—	—	—	—	PCC0
PCPU	—	—	—	—	—	—	—	PCPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PDWU	PDWU7	PDWU6	PDWU5	PDWU4	PDWU3	PDWU2	PDWU1	PDWU0
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0

Register Name	Bit							
	7	6	5	4	3	2	1	0
PF	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0

"—": Unimplemented, read as "0"

I/O Logic Function Register List – HT68FB571

Pull-high Resistors

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

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

• PxPU Register

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

PxPUn: I/O port x pin pull-high function control
0: Disable
1: Enable

The PxPUn bit is used to control the pin pull-high function. Here the "x" can be A, B, C, D, E or F respectively depending upon the selected device. However, the actual available bits for each I/O Port may be different.

I/O 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 I/O pins with wake up ability from high to low. This function is especially suitable for applications that can be woken up via external switches. Each I/O pin with wake-up ability can be selected individually to have this wake-up feature using the PxWU register.

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

• PxWU Register

Bit	7	6	5	4	3	2	1	0
Name	PxWU7	PxWU6	PxWU5	PxWU4	PxWU3	PxWU2	PxWU1	PxWU0
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

PxWUn: I/O port x pin wake-up function control
0: Disable
1: Enable

The PxWUn bit is used to control the pin wake-up function. Here the "x" can be A, C or D respectively depending upon the selected device. However, the actual available bits for each I/O Port may be different.

I/O Port Control Registers

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

• PxC Register

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

PxCn: I/O port x pin type selection
0: Output
1: Input

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

I/O Port Power Source Control

The devices support different I/O port power source selections for several I/O pins, namely the PA0, PA2, PB1~PB3 pins for the HT68FB541 device and the PA1, PA3, PB4~PB7 pins for the HT68FB571 device. The port power can come from the power pin VDD, VDDIO or V33O, which is determined using the PMPS bit field in the PMPS register. The VDDIO power pin function should first be selected using the corresponding pin-shared function selection bits if the port power is supposed to come from the VDDIO pin. An important point to know is that the input power voltage on the VDDIO pin should be equal to or less than the device supply power voltage when the VDDIO pin is selected as the port power supply pin. With the exception of $\overline{\text{RES}}$/OCDS, the multi-power function is only effective when the pin is set to have digital input or output function.

• PMPS Register – HT68FB541

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PMPS3	PMPS2	PMPS1	PMPS0
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 **PMPS3~PMPS2**: PB3~PB1 power supply selection
00: V_{DD}
01: V_{DD}
10: V_{DDIO}
11: V_{V33O} – 3.3V regulator output

Bit 1~0 **PMPS1~PMPS0**: PA2, PA0 power supply selection
00: V_{DD}
01: V_{DD}
10: V_{DDIO}
11: V_{V33O} – 3.3V regulator output

• **PMPS Register – HT68FB571**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PMPS3	PMPS2	PMPS1	PMPS0
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 **PMPS3~PMPS2**: PB7~PB4 power supply selection
00: V_{DD}
01: V_{DD}
10: V_{DDIO}
11: V_{V33O} – 3.3V regulator output

Bit 1~0 **PMPS1~PMPS0**: PA3, PA1 power supply selection
00: V_{DD}
01: V_{DD}
10: V_{DDIO}
11: V_{V33O} – 3.3V regulator output

I/O Port Source Current Selection

The devices support different source current driving capability for several I/O pins. With the corresponding selection register, SLEDCn, the specific I/O pins can support four levels of the source current driving capability. Note that for the HT68FB571, there is no serial number "n" in the relevant register and control bits since there is only one control register in the device. Users should refer to the Input/Output Characteristics section to select the desired source current for different applications.

• **SLEDC0 Register – HT68FB541**

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

Bit 7~6 **SLEDC07~SLEDC06**: PA7 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 5~4 **SLEDC05~SLEDC04**: PA6 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 3~2 **SLEDC03~SLEDC02**: PA3 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 1~0 **SLEDC01~SLEDC00**: PA1 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

• SLEDC1 Register – HT68FB541

Bit	7	6	5	4	3	2	1	0
Name	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
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 **SLEDC17~SLEDC16**: PF1 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 5~4 **SLEDC15~SLEDC14**: PF0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 3~2 **SLEDC13~SLEDC12**: PB5 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 1~0 **SLEDC11~SLEDC10**: PB4 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

• SLEDC Register – HT68FB571

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SLEDC3	SLEDC2	SLEDC1	SLEDC0
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 **SLEDC3~SLEDC2**: PF7~PF4 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 1~0 **SLEDC1~SLEDC0**: PF3~PF0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

I/O Port Input Voltage Level Selection – HT68FB571 Only

The Port D pins in the HT68FB571 device can be configured with different input voltage level which is selected by the corresponding pin input voltage level select bits. Users should refer to the Input/Output Characteristics section to obtain the exact value for different applications.

- **INLVC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INHLVC1	INHLVC0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1 **INHLVC1**: PD7~PD4 input V_{IL} & V_{IH} selection
0: V_{IL} & V_{IH} selection=Level 0
1: V_{IL} & V_{IH} selection=Level 1

Bit 0 **INHLVC0**: PD3~PD0 input V_{IL} & V_{IH} selection
0: V_{IL} & V_{IH} selection=Level 0
1: V_{IL} & V_{IH} selection=Level 1

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	—	—	PAS03	PAS02	—	—
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	—	—	—	—	PBS13	PBS12	PBS11	PBS10
PCS0	—	—	—	—	—	—	PCS01	PCS00
PFS0	—	—	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00

Pin-shared Function Selection Register List – HT68FB541

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	—	—	PAS03	PAS02	—	—
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	—	—	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	—	—	—	—	—	—	PCS01	PCS00
PDS0	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
PDS1	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
PES0	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
PES1	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
PFS0	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
PFS1	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10

Pin-shared Function Selection Register List – HT68FB571

- **PAS0 Register – HT68FB541**

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

Bit 7~6 **PAS07~PAS06**: PA3 pin-shared function selection
00/01/10: PA3
11: KEYR1/LEDCOM1

Bit 5~4 Unimplemented, read as "0"

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00/01/10: PA1
11: KEYR0/LEDCOM0

Bit 1~0 Unimplemented, read as "0"

- **PAS0 Register – HT68FB571**

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

Bit 7~6 **PAS07~PAS06**: PA3 pin-shared function selection
00: PA3
01: PFD0
10: SPISDI
11: KEYR16

Bit 5~4 Unimplemented, read as "0"

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00/01: PA1/TC0
10: SPISDO
11: KEYR17

Bit 1~0 Unimplemented, read as "0"

• **PAS1 Register – HT68FB541**

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

Bit 7~6 **PAS17~PAS16**: PA7 pin-shared function selection
00/01: PA7
10: CMP0INN
11: KEYR6/LEDCOM6

Bit 5~4 **PAS15~PAS14**: PA6 pin-shared function selection
00/01: PA6
10: CMP0INP
11: KEYR7/LEDCOM7

Bit 3~2 **PAS13~PAS12**: PA5 pin-shared function selection
00/01/10: PA5
11: LEDSEG0

Bit 1~0 **PAS11~PAS10**: PA4 pin-shared function selection
00/01/10: PA4
11: LEDSEG1

• **PAS1 Register – HT68FB571**

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

Bit 7~6 **PAS17~PAS16**: PA7 pin-shared function selection
00/01/10: PA7
11: KEYR3/LEDCOM3

Bit 5~4 **PAS15~PAS14**: PA6 pin-shared function selection
00/01/10: PA6
11: KEYR2/LEDCOM2

Bit 3~2 **PAS13~PAS12**: PA5 pin-shared function selection
00/01/10: PA5
11: KEYR1/LEDCOM1

Bit 1~0 **PAS11~PAS10**: PA4 pin-shared function selection
00/01/10: PA4
11: KEYR0/LEDCOM0

• **PBS0 Register – HT68FB541**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PBS07~PBS06**: PB3 pin-shared function selection
00/01/10: PB3/TC0
11: SPISDO

Bit 5~4 **PBS05~PBS04**: PB2 pin-shared function selection
00/01: PB2
10: PFD0
11: SPISDI

Bit 3~2 **PBS03~PBS02**: PB1 pin-shared function selection
00/01: PB1/INT1
10: PFD1
11: SPISCK

Bit 1~0 **PBS01~PBS00**: PB0 pin-shared function selection
00/01: PB0/TC1
10: $\overline{\text{SPISCS}}$
11: VDDIO

• **PBS0 Register – HT68FB571**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PBS07~PBS06**: PB3 pin-shared function selection
00/01/10: PB3
11: KEYR7/LEDCOM7

Bit 5~4 **PBS05~PBS04**: PB2 pin-shared function selection
00/01/10: PB2
11: KEYR6/LEDCOM6

Bit 3~2 **PBS03~PBS02**: PB1 pin-shared function selection
00/01/10: PB1
11: KEYR5/LEDCOM5

Bit 1~0 **PBS01~PBS00**: PB0 pin-shared function selection
00/01/10: PB0
11: KEYR4/LEDCOM4

• **PBS1 Register – HT68FB541**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **PBS13~PBS12**: PB5 pin-shared function selection
00/01: PB5
10: CMP1INP
11: KEYR5/LEDCOM5

Bit 1~0 **PBS11~PBS10**: PB4 pin-shared function selection
00/01: PB4
10: CMP1INN
11: KEYR4/LEDCOM4

• **PBS1 Register – HT68FB571**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PBS15~PBS14**: PB6 pin-shared function selection
00/01/10: PB6/INT0
11: PFD1

Bit 3~2 **PBS13~PBS12**: PB5 pin-shared function selection
00/01: PB5/TC1
10: $\overline{\text{SPISCS}}$
11: KEYR19

Bit 1~0 **PBS11~PBS10**: PB4 pin-shared function selection
00/01: PB4/INTI
10: SPISCK
11: KEYR18

• **PCS0 Register – HT68FB541**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PCS01~PCS00**: PC0 pin-shared function selection
00/01/10: PC0/$\overline{\text{RES}}$
11: KEYC0

• **PCS0 Register – HT68FB571**

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

Bit 7~2 Unimplemented, read as "0"

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

• **PDS0 Register – HT68FB571**

Bit	7	6	5	4	3	2	1	0
Name	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
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 **PDS07~PDS06**: PD3 pin-shared function selection
00/01/10: PD3
11: KEYC3

Bit 5~4 **PDS05~PDS04**: PD2 pin-shared function selection
00/01/10: PD2
11: KEYC2

Bit 3~2 **PDS03~PDS02**: PD1 pin-shared function selection
00/01/10: PD1
11: KEYC1

Bit 1~0 **PDS01~PDS00**: PD0 pin-shared function selection
00/01/10: PD0
11: KEYC0

• **PDS1 Register – HT68FB571**

Bit	7	6	5	4	3	2	1	0
Name	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
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 **PDS17~PDS16**: PD7 pin-shared function selection
00/01/10: PD7
11: KEYC7

Bit 5~4 **PDS15~PDS14**: PD6 pin-shared function selection
00/01/10: PD6
11: KEYC6

Bit 3~2 **PDS13~PDS12**: PD5 pin-shared function selection
00/01/10: PD5
11: KEYC5

Bit 1~0 **PDS11~PDS10**: PD4 pin-shared function selection
00/01/10: PD4
11: KEYC4

• **PES0 Register – HT68FB571**

Bit	7	6	5	4	3	2	1	0
Name	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
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 **PES07~PES06**: PE3 pin-shared function selection
00/01/10: PE3
11: KEYR11/LEDCOM11

Bit 5~4 **PES05~PES04**: PE2 pin-shared function selection
00/01/10: PE2
11: KEYR10/LEDCOM10

Bit 3~2 **PES03~PES02**: PE1 pin-shared function selection
00/01/10: PE1
11: KEYR9/LEDCOM9

Bit 1~0 **PES01~PES00**: PE0 pin-shared function selection
00/01/10: PE0
11: KEYR8/LEDCOM8

• **PES1 Register – HT68FB571**

Bit	7	6	5	4	3	2	1	0
Name	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
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 **PES17~PES16**: PE7 pin-shared function selection
00/01/10: PE7
11: KEYR15/LEDCOM15

Bit 5~4 **PES15~PES14**: PE6 pin-shared function selection
00/01/10: PE6
11: KEYR14/LEDCOM14

Bit 3~2 **PES13~PES12**: PE5 pin-shared function selection
00/01/10: PE5
11: KEYR13/LEDCOM13

Bit 1~0 **PES11~PES10**: PE4 pin-shared function selection
00/01/10: PE4
11: KEYR12/LEDCOM12

• PFS0 Register – HT68FB541

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PFS05~PFS04**: PF2 pin-shared function selection
00/01/10: PF2
11: LEDSEG2

Bit 3~2 **PFS03~PFS02**: PF1 pin-shared function selection
00/01: PF1
10: CMP1O
11: KEYR3/LEDCOM3

Bit 1~0 **PFS01~PFS00**: PF0 pin-shared function selection
00/01: PF0
10: CMP0O
11: KEYR2/LEDCOM2

• PFS0 Register – HT68FB571

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

Bit 7~6 **PFS07~PFS06**: PF3 pin-shared function selection
00/01/10: PF3
11: LEDSEG3

Bit 5~4 **PFS05~PFS04**: PF2 pin-shared function selection
00/01/10: PF2
11: LEDSEG2

Bit 3~2 **PFS03~PFS02**: PF1 pin-shared function selection
00/01/10: PF1
11: LEDSEG1

Bit 1~0 **PFS01~PFS00**: PF0 pin-shared function selection
00/01/10: PF0
11: LEDSEG0

• **PFS1 Register – HT68FB571**

Bit	7	6	5	4	3	2	1	0
Name	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PFS17~PFS16**: PF7 pin-shared function selection
00/01/10: PF7
11: LEDSEG7

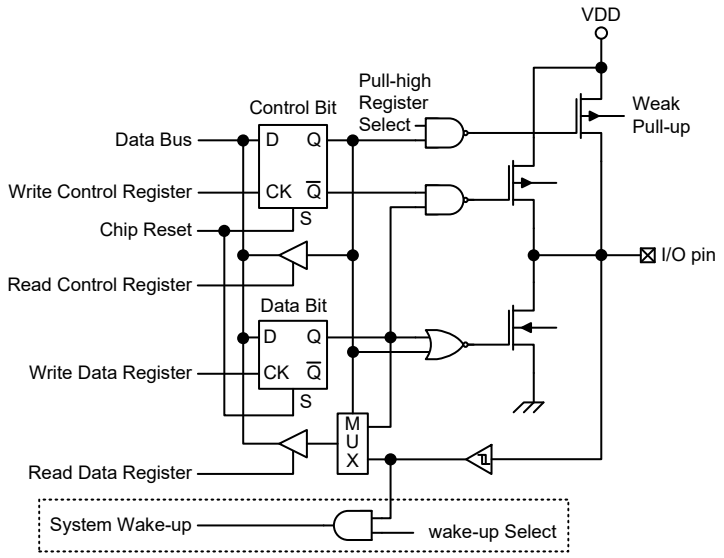
Bit 5~4 **PFS15~PFS14**: PF6 pin-shared function selection
00/01/10: PF6
11: LEDSEG6

Bit 3~2 **PFS13~PFS12**: PF5 pin-shared function selection
00/01/10: PF5
11: LEDSEG5

Bit 1~0 **PFS11~PFS10**: PF4 pin-shared function selection
00/01/10: PF4
11: LEDSEG4

I/O Pin Structures

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



Logic Function Input/Output Structure

Programming Considerations

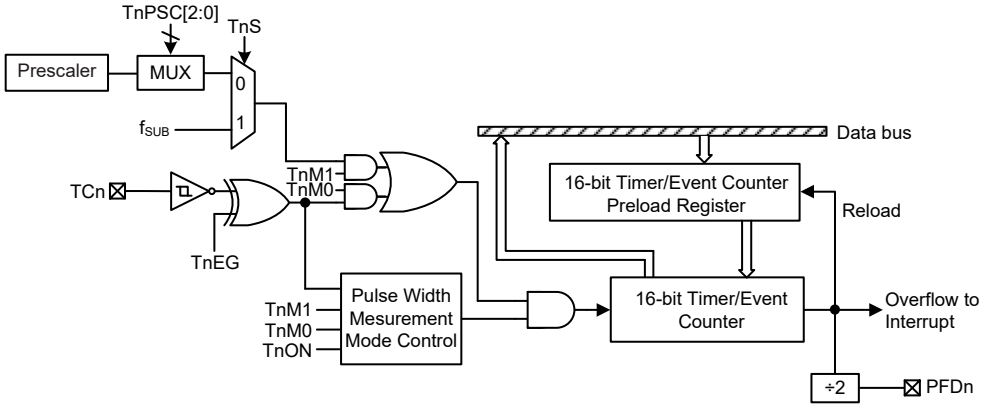
Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers are first programmed. Selecting which pins are inputs and which are outputs can be

achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

Several I/O pins have the additional capability of providing wake-up function, namely the Port A and PC0 pin on the HT68FB541 device and the Port D on the HT68FB571 device. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition on the I/O pins with wake-up ability. Single or multiple pins on these specific ports can be setup to have this function.

Timer/Event Counter

The provision of timer/event counter forms an important part of any microcontroller, giving the designer a means of carrying out time related functions. Each of the devices contains two 16-bit up-counting timer/event counters. As the timer/event counter has three different operating modes, it can be configured to operate as a general timer, an external event counter or as a pulse width measurement device. The provision of an internal prescaler to the clock circuitry gives added range to the timer.



16-bit Timer/Event Counter Structure (n=0 or 1)

Timer/Event Counter Input Clock Source

The timer/event counter clock source can originate from various sources, an internal clock or an external pin. The internal clock source is used when the timer is in the Timer mode and Pulse Width Measurement mode. For the timer/event counter, this internal clock source can be configured by the TnS bit in the register TMRnC to be derived from a prescaler, the division ratio of which is configured by the TnPSC2~TnPSC0 bits in the Timer Control Register TMRnC, or to be derived from the f_{SUB} clock.

An external clock source is used when the timer/event counter is in the Event Counter mode, the clock source is provided on the external timer pin TCn. Depending upon the condition of the TnEG bit, each high to low or low to high transition on the external timer pin will increase the counter by one.

Timer/Event Counter Registers

There are two types of registers related to the timer/event counter. The first is the register pair that contains the actual value of the timer and into which an initial value can be preloaded. Reading from this register pair retrieves the contents of the timer/event counter. The second type of associated register is the Timer Control Register which defines the timer options and determines how the timer is to be used. The device can have the timer clock configured to come from the internal clock source or the external timer pin.

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMRnC	TnM1	TnM0	TnS	TnON	TnEG	TnPSC2	TnPSC1	TnPSC0
TMRnL	D7	D6	D5	D4	D3	D2	D1	D0
TMRnH	D15	D14	D13	D12	D11	D10	D9	D8

Timer/Event Counter Register List (n=0 or 1)

Timer Register – TMRnL, TMRnH

As each timer/event counter contains an internal 16-bit timer, it requires two data registers to store the value. There are a high byte register, known as TMRnH, and a low byte register, known as TMRnL. The value in the timer register pair increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFFFH, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reloaded with the initial preload register value and continue counting.

Note that to achieve a maximum full counting range of FFFFH, the preload register must first be cleared. If the timer/event counter is in an OFF condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will be remained in the preload register and will not be written into the actual counter until an overflow occurs.

Writing to the TMRnL register will only write the data into an internal lower byte buffer while writing to the TMRnH register will transfer the high byte data and the contents of the lower byte buffer into the TMRnH and TMRnL registers respectively. Therefore the timer/event counter preload register is changed by each write operation to the TMRnH register. Reading from the TMRnH register will latch the contents of the TMRnH and TMRnL counters to the destination and the lower byte buffer respectively, while reading from TMRnL will read the contents of the lower byte buffer.

- **TMRnL Register**

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

Bit 7~0 **D7~D0**: Timer preload register low byte

- **TMRnH register**

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

Bit 7~0 **D15~D8**: Timer preload register high byte

Timer Control Register – TMRnC

The Timer Control Register is known as TMRnC. It is the Timer Control Register together with its corresponding timer registers that controls the full operation of the timer/event counter. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

To select which of the three modes the timer is to operate in, namely the Timer mode, the Event Counter mode or the Pulse Width Measurement mode, the TnM1/TnM0 bits in the Timer Control Register must be set to the required logic levels. The timer-on bit TnON provides the basic on/off control of the respective timer. Setting the bit to high allows the counter to run. Clearing the bit stops the counter. When the internal clock source is used, it can be sourced from the prescaler clock or the clock f_{SUB} selected by setting the TnS bit. Bits TnPSC2~TnPSC0 determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the Event Counter or Pulse Width Measurement Mode, the active transition edge level type is selected by the logic level of the TnEG bit in the Timer Control Register.

- **TMR0C Register**

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	T0S	T0ON	T0EG	T0PSC2	T0PSC1	T0PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7~6 **T0M1~T0M0**: Timer/event counter 0 operation mode selection
00: Unused
01: Event Counter mode
10: Timer Mode
11: Pulse Width Measurement mode

Bit 5 **T0S**: Timer/event counter 0 internal clock source selection
0: From the prescaler clock
1: From f_{SUB}

Bit 4 **T0ON**: Timer/event counter 0 counting enable control
0: Disable
1: Enable

Bit 3 **T0EG**: Timer/event counter 0 active edge selection
Event Counter mode
0: Count on rising edge
1: Count on falling edge
Pulse Width Measurement mode
0: Start counting on falling edge, stop on rising edge
1: Start counting on rising edge, stop on falling edge

Bit 2~0 **T0PSC2~T0PSC0**: Timer/event counter 0 prescaler rate selection
000: f_{PSC}
001: $f_{PSC}/2$
010: $f_{PSC}/4$
011: $f_{PSC}/8$
100: $f_{PSC}/16$
101: $f_{PSC}/32$
110: $f_{PSC}/64$
111: $f_{PSC}/128$

• **TMR1C Register**

Bit	7	6	5	4	3	2	1	0
Name	T1M1	T1M0	T1S	T1ON	T1EG	T1PSC2	T1PSC1	T1PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7~6 **T1M1~T1M0**: Timer/event counter 1 operation mode selection
00: Unused
01: Event Counter mode
10: Timer Mode
11: Pulse Width Measurement mode

Bit 5 **T1S**: Timer/event counter 1 internal clock source selection
0: From the prescaler clock
1: From f_{SUB}

Bit 4 **T1ON**: Timer/event counter 1 counting enable control
0: Disable
1: Enable

Bit 3 **T1EG**: Timer/event counter 1 active edge selection
Event Counter mode
0: Count on rising edge
1: Count on falling edge
Pulse Width Measurement mode
0: Start counting on falling edge, stop on the rising edge
1: Start counting on rising edge, stop on the falling edge

Bit 2~0 **T1PSC2~T1PSC0**: Timer/event counter 1 prescalar rate selection
000: $f_{PSC}/16$
001: $f_{PSC}/32$
010: $f_{PSC}/64$
011: $f_{PSC}/128$
100: $f_{PSC}/256$
101: $f_{PSC}/512$
110: $f_{PSC}/1024$
111: $f_{PSC}/2048$

Timer/Event Counter Operating Modes

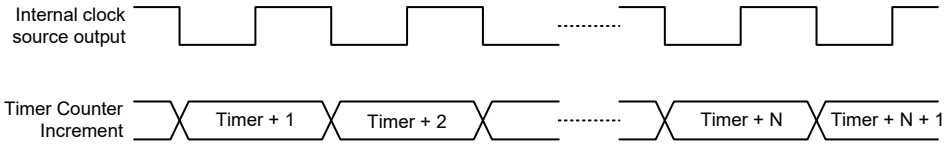
The timer/event counter can operate in one of three operating modes, Timer mode, Event Counter mode or Pulse Width Measurement mode. The operating mode is selected using the TnM1 and TnM0 bits in the TMRnC register.

Timer Mode

To select this mode, bits TnM1 and TnM0 in the TMRnC register should be set to 10 respectively. In this mode, the timer/event counter can be utilized to measure fixed time intervals, providing an internal interrupt signal each time the timer/event counter overflows.

When operating in this mode, the internal clock is used as the timer clock. If the TnS bit in the TMRnC register is zero, the timer input clock source can be f_{SYS}, $f_{SYS}/4$ or f_{SUB}, determined by the PSCR register. However, this timer clock source will be further divided by a prescaler, the value of which is determined by the bits TnPSC2~TnPSC0 in the Timer Control Register. If the TnS bit is high, then the internal clock is sourced from the f_{SUB} clock. The timer-on bit, TnON must be set high to enable the timer. Each time an internal clock high to low transition occurs, the timer increases by one. It should be noted that in the Timer mode, even if the device is in the IDLE/SLEEP mode, if the selected internal clock is still activated the timer will continue to count. When the timer reaches its maximum 16-bit, FFFF Hex, value and overflows, an interrupt request is generated and the

timer will reload the value already loaded into the preload register and continue counting. A timer overflow condition will trigger the corresponding internal interrupt request, with the TnF flag being set, which is one of wake-up sources. In this condition, if the corresponding interrupt enable bit TnE in the INTC2 register has been set, a Timer/Event Counter Interrupt will be generated.



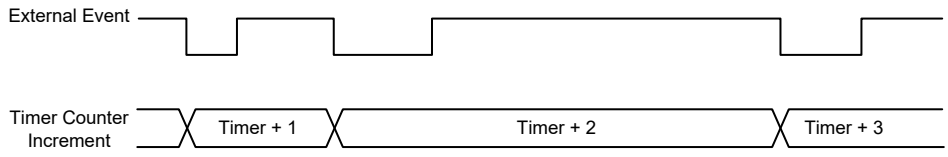
Timer Mode Timing Chart

Event Counter Mode

To select this mode, bits TnM1 and TnM0 in the TMRnC register should be set to 01 respectively. In this mode, a number of externally changing logic events, occurring on the external timer TCn pin, can be recorded by the timer/event counter.

When operating in this mode, the external timer TCn pin is used as the timer/event counter clock source. After the other bits in the Timer Control Register have been properly set, the enable bit TnON in the Timer Control Register, can be set high to enable the timer/event counter. If the Active Edge Selection bit, TnEG, is low, the Timer/Event Counter will increase by one each time the external timer pin receives a low to high transition. If the TnEG is high, the counter will increase by one each time the external timer pin receives a high to low transition. When it is full and overflows, an interrupt request is generated and the timer/event counter will reload the value already loaded into the preload register and continue counting. However, the internal interrupts can be disabled by ensuring that the TnE bit of the INTC2 register is reset to zero.

As the external timer pin TCn is pin-shared with other functions, the TCn pin function must first be setup using relevant pin-shared function selection register and should also be set as an input by setting the corresponding bit in the port control register. It should be noted that in the Event Counter mode, even if the microcontroller is in the IDLE/SLEEP Mode, the timer/event counter will continue to record externally changing logic events on the timer input pin TCn. As a result when the timer overflows it will generate a timer interrupt request which can be a wake-up source.



Event Counter Mode Timing Chart (TnEG=1)

Pulse Width Measurement Mode

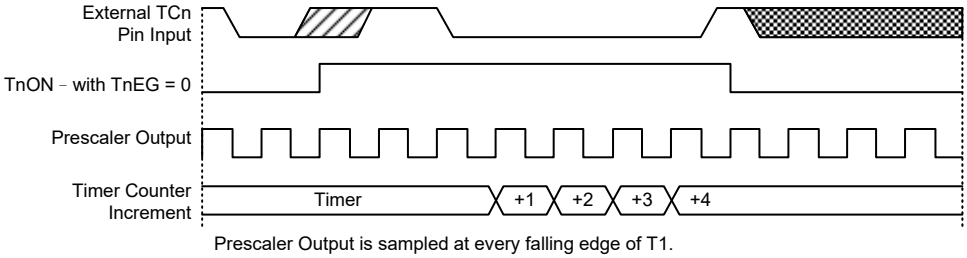
To select this mode, bits TnM1 and TnM0 in the TMRnC register should be set to 11 respectively. In this mode, the timer/event counter can be utilised to measure the width of external pulses applied to the external timer pin.

When operating in this mode, the internal clock is used as the timer clock. If the TnS bit in the TMRnC register is zero, the timer input clock source can be f_{SYS}, $f_{SYS}/4$ or f_{SUB}, determined by the PSCR register. However, this timer clock source will be further divided by a prescaler, the value of which is determined by the bits TnPSC2~TnPSC0 in the Timer Control Register. If the TnS bit is high, then the internal clock is sourced from the f_{SUB} clock. As the external timer pin TCn is pin-shared with other functions, the TCn pin function must first be setup using relevant pin-shared function selection register and should also be set as an input by setting the corresponding bit in

the port control register. After other bits in the Timer Control Register have been properly set, the enable bit TnON can be set high to enable the timer/event counter, however it will not actually start counting until an active edge is received on the external timer pin.

If the Active Edge Selection bit TnEG is low, once a high to low transition has been received on the external timer pin, the timer/event counter will start counting based on the internal selected clock source, until the external timer pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the timer/event counter will stop counting. If the Active Edge Select bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. The enable bit will then be automatically reset to zero and the timer/event counter will stop counting. It is important to note that in the Pulse Width Measurement mode, the enable bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under application program control.

The residual value in the timer/event counter, which can now be read by the program, therefore represents the length of the pulse received on the TCn pin. As the enable bit has now been reset, any further transitions on the external timer pin will be ignored. The timer cannot begin further pulse width measurement until the enable bit is set high again by the application program. In this way, single shot pulse measurement can be easily made. It should be noted that in this mode the timer/event counter is controlled by logical transitions on the external timer pin and not by the logic level. When the timer/event counter is full and overflows, an interrupt request is generated and the timer/event counter will reload the value already loaded into the preload register and continue counting. It should be noted that in the Pulse Width Measurement mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to increase if the selected internal clock source is still activated and the required logical transitions occur on the external TCn pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit TnE in the INTC2 register is reset to zero.



Pulse Width Measurement Mode Timing Chart (TnEG=0)

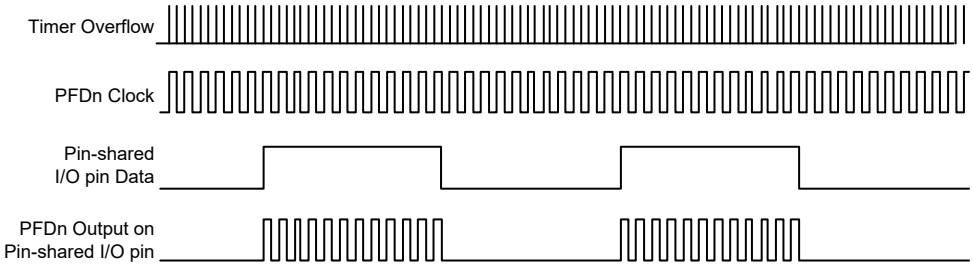
Prescaler

Bits TnPSC2~TnPSC0 of the TMRnC register can be used to define a division ratio for the internal clock source of the timer/event counter. It should be noted that as the timer/event counter internal clock source is the same as the Time Base clock source, care should be taken when programming. For more detailed information concerning the prescaler, refer to the Time Base Interrupt section.

PFD Function

The Programmable Frequency Divider provides a means of producing a variable frequency output suitable for application, such as some interfaces requiring a precise frequency generator. As the PFDn output pin is pin-shared with other functions, the PFDn pin function must first be setup using relevant pin-shared function selection register.

The timer/event counter overflow signal is the clock source for the PFD function. The output frequency is controlled by loading the required values into the timer prescaler and timer registers to give the required division ratio. The counter will begin to count-up from this preload register value until full, at which point an overflow signal is generated, causing the PFDn output to change state simultaneously. Then the counter will be automatically reloaded with the preload register value and continue counting.



PFD Function

Programming Considerations

When running in the Timer mode, the internal system clock is used as the timer clock source and is therefore synchronised with the overall operation of the microcontroller. In this mode when the timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector. For the Pulse Width Measurement mode, the internal system clock is also used as the timer clock source but the timer will only run when the correct logic condition appears on the external timer input pin. As this is an external event and not synchronised with the internal timer clock, the microcontroller will only see this external event when the next timer clock pulse arrives. As a result, there may be small differences in measured values requiring programmers to take this into account during programming. The same applies if the timer is configured to operate in the Event Counter mode, which again is an external event and not synchronised with the internal system or timer clock.

When the timer/event counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, it should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialised before using them for the first time. The associated timer interrupt enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The active edge selection, timer operating mode selection and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application. It is also important to ensure that a desired initial value is first loaded into the timer registers before the timer is switched on. After the timer has been initialised the timer can be turned on and off by controlling the enable bit in the timer control register.

When the timer/event counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the timer/event counter interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a timer/event counter overflow will also generate a wake-up signal if the device is in the IDLE/SLEEP mode. This situation may occur if the timer/event counter internal clock is still on or if the external signal continues to change state. In such cases, the timer/event counter will continue to count and if an overflow occurs the device will be woken up. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the “HALT” instruction to enter the IDLE/SLEEP mode.

Serial Peripheral Interface – SPI

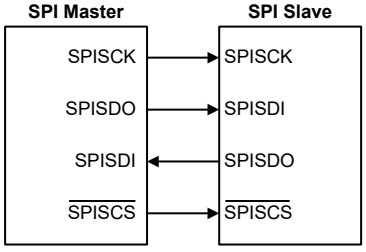
The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four-line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the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

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

SPI Interface Operation

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

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

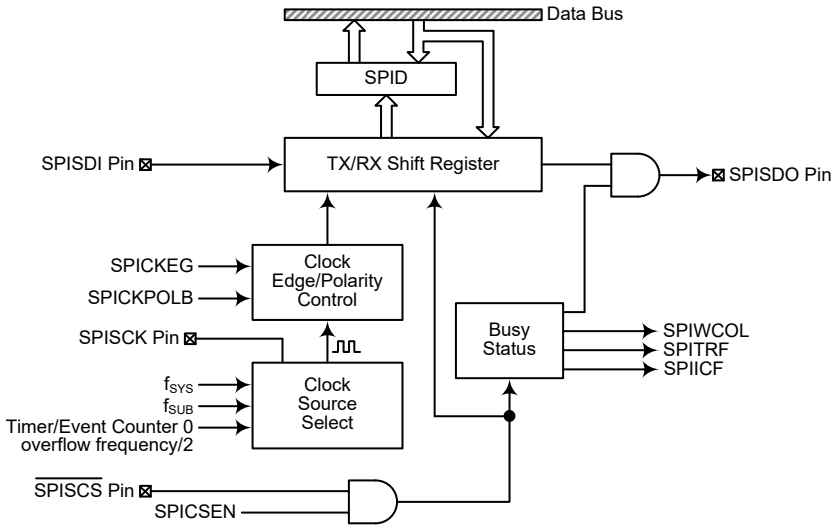


SPI Master/Slave Connection

The SPI function in these devices offers the following features:

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

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



SPI Block Diagram

SPI Registers

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

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

SPI Register List

SPI Data Register

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

- **SPID Register**

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

"x": Unknown

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

SPI Control Registers

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

• **SPIC0 Register**

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

Bit 7~5 **SPIM2~SPIM0**: SPI Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is Timer/Event Counter 0 overflow frequency/2
101: SPI slave mode
110: SPI disabled
111: SPI disabled
These bits are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from timer/event counter 0 and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4~2 Unimplemented, read as "0"

Bit 1 **SPIEN**: SPI Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SPI interface. When the SPIEN bit is cleared to zero to disable the SPI interface, the SPISDI, SPISDO, SPISCK and $\overline{\text{SPISCS}}$ lines will lose their SPI function and the SPI operating current will be reduced to a minimum value. When the bit is high the SPI interface is enabled.

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

• **SPIC1 Register**

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

Bit 7~6 Unimplemented, read as "0"

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

Bit 4 **SPICKEG**: SPI SPISCK clock active edge type selection
SPICKPOLB=0
0: SPISCK has high base level with data capture on SPISCK rising edge
1: SPISCK has high base level with data capture on SPISCK falling edge

SPICKPOLB=1
0: SPISCK has low base level with data capture on SPISCK falling edge
1: SPISCK has low base level with data capture on SPISCK rising edge

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

Bit 3 **SPIMLS**: SPI data shift order
0: LSB first
1: MSB first

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

Bit 2 **SPICSEN**: SPI $\overline{\text{SPISCS}}$ pin control
0: Disable
1: Enable

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

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

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

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

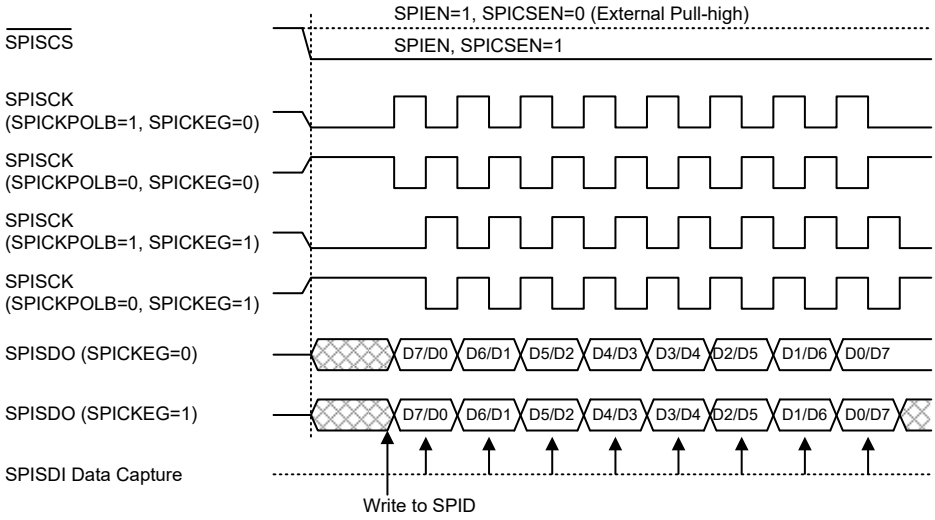
The SPITRF bit is the Transmit/Receive Complete flag and is set "1" automatically when an SPI data transmission is completed, but must set to zero by the application program. It can be used to generate an interrupt.

SPI Communication

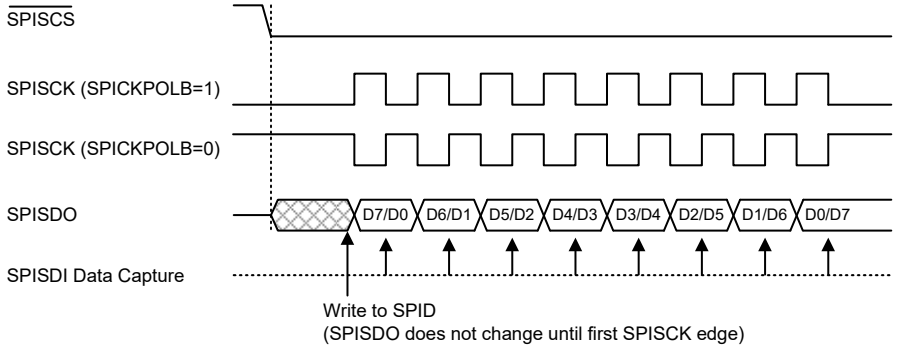
After the SPI interface is enabled by setting the SPIEN bit high, then in the Master Mode, when data is written to the SPID register, transmission/reception will begin simultaneously. When the data transfer is complete, the SPITRF flag will be set automatically, but must be cleared using the application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

The master should output an $\overline{\text{SPISCS}}$ 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 SPISCK signal depending upon the configurations of the SPICKPOLB bit and SPICKEG bit. The accompanying timing diagram shows the relationship between the slave data and SPISCK signal for various configurations of the SPICKPOLB and SPICKEG bits.

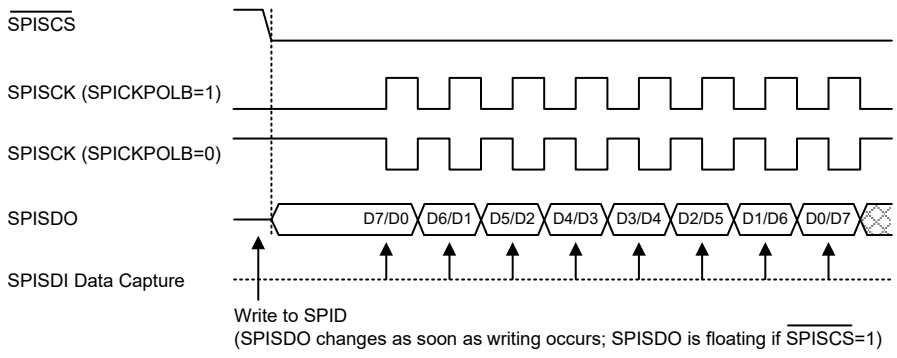
The SPI will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.



SPI Master Mode Timing

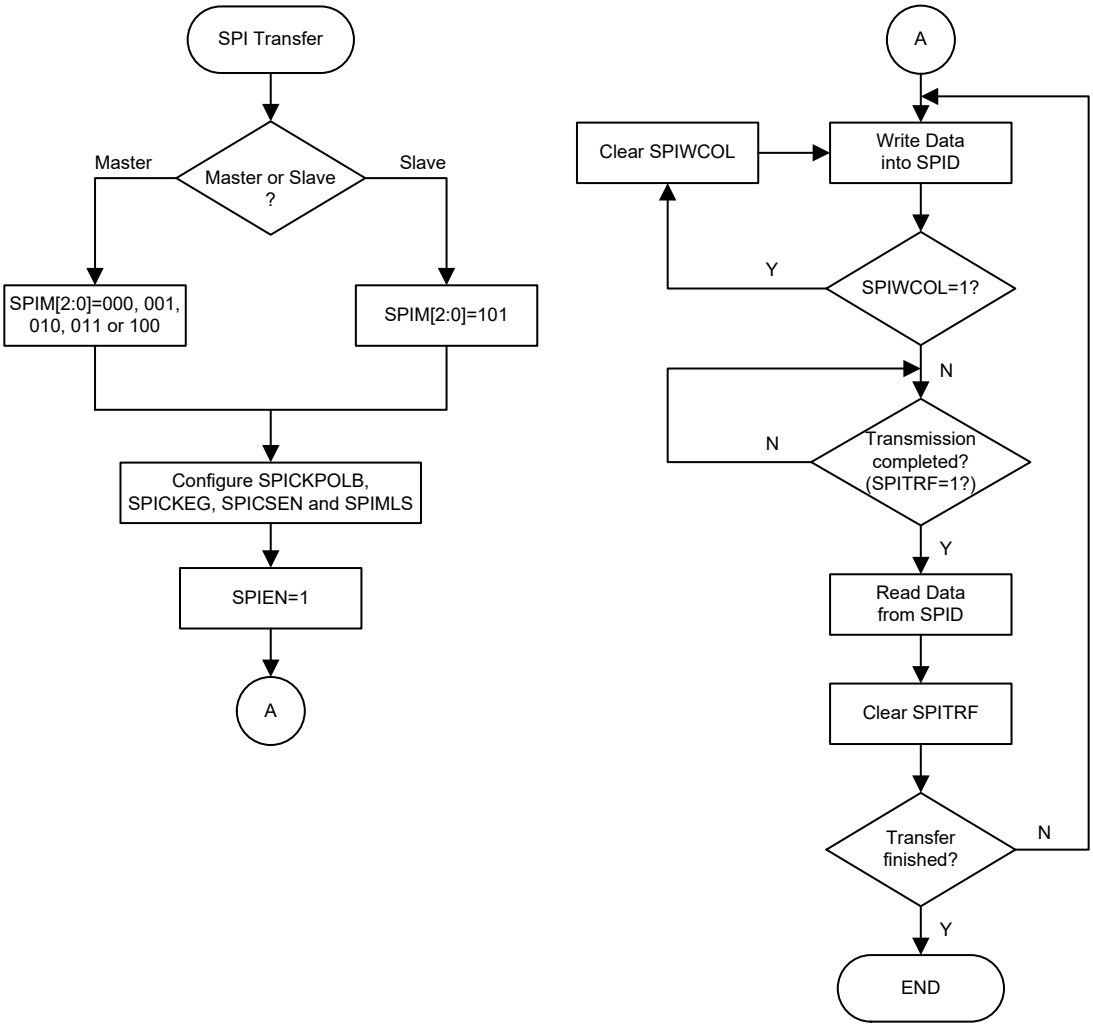


SPI Slave Mode Timing – SPICKEG=0



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

SPI Slave Mode Timing – SPICKEG=1



SPI Transfer Control Flowchart

SPI Bus Enable/Disable

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

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

SPI Operation

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

The SPICSEN bit in the SPIC1 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{\text{SPISCS}}$ line to be active, which can then be used to control the SPI interface. If the SPICSEN bit is low, the SPI interface will be disabled and the $\overline{\text{SPISCS}}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the SPICSEN bit and the SPIEN bit in the SPIC0 register are set high, this will place the

SPISDI line in a floating condition and the SPISDO line high. If in Master Mode the SPISCK line will be either high or low depending upon the clock polarity selection bit SPICKPOLB in the SPIC1 register. If in Slave Mode the SPISCK line will be in a floating condition. If SPIEN is low then the bus will be disabled and $\overline{\text{SPISCS}}$, SPISDI, SPISDO and SPISCK will all become I/O pins or the other functions. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SPID register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode

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

Slave Mode

- Step 1
  Select the SPI Slave mode using the SPIM2~SPIM0 bits in the SPIC0 control register.
- Step 2
  Setup the SPICSEN bit and setup the SPIMLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master device.
- Step 3
  Setup the SPIEN bit in the SPIC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SPID register, which will actually place the data into the TXRX buffer. Then wait for the master clock SPISCK and $\overline{\text{SPISCS}}$ signal. After this, go to

step 5.

For read operations: the data transferred in on the SPISDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SPID register.

- Step 5
  Check the SPIWCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the SPITRF bit or wait for an SPI serial bus interrupt.
- Step 7
  Read data from the SPID register.
- Step 8
  Clear SPITRF.
- Step 9
  Go to step 4.

Error Detection

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

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.

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

Power Plane

There are three power planes for the devices and they are USB SIE VDD, VDDIO and the MCU VDD.

For the USB SIE VDD it will supply power for all circuits related to USB SIE and is sourced from the UBUS pin. Once the USB is removed from the USB interface and there is no power in the USB BUS, the USB SIE circuit is no longer operational.

For the PA0, PA2, PB1~PB3 pins on the HT68FB541 device and the PA1, PA3 and PB4~PB7 pins on the HT68FB571 device, the power can be supplied by the VDD, V33O or VDDIO pin selected using the PMPS register.

The VDDIO is pin-shared with PB0 or PC0 depending upon the selected device. The VDDIO function can be selected by the corresponding pin-shared function selection bits.

For the MCU VDD, it supplies power for all the device circuits except the USB SIE which is supplied by UBUS.

USB Interface Operation

To communicate with an external USB host, the internal USB module has the external pins known as UDP and UDN along with the 3.3V regulator output V33O. A Serial Interface Engine (SIE) decodes the incoming USB data stream and transfers it to the correct endpoint buffer memory known as the FIFO. In these devices, the USB module has 4 endpoints, EP0 ~ EP3. Endpoint 0 has 8-byte size, Endpoint 1 has 16-byte size, Endpoint 2 has 32-byte size and Endpoint 3 has 64-byte size. The endpoint 0 supports the Control transfer while the endpoint 1 ~ endpoint 3 support the Interrupt or Bulk transfer.

USB Interface Registers

The USB function control is implemented using a series of registers. A series of status registers provide the user with the USB data transfer situation as well as any error conditions. The USB contains its own independent interrupt which can be used to indicate when the USB FIFOs are accessed by the host device or a change of the USB operating conditions including the USB suspend mode, resume event or USB reset occurs.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SYSC	BITWE	USBDIS	RUBUS	UBUSF	—	—	D1	ESDF
USB_STAT	OD1O	OD0O	OD1I	OD0I	SE1	SE0	PU	RUPH
UINT	—	—	—	—	EP3EN	EP2EN	EP1EN	EP0EN
USC	URD	UMS2	UMS1	UMS0	RESUME	URST	RMWK	SUSP
UESR	—	—	—	—	EP3F	EP2F	EP1F	EP0F
UCC	RCTRL	—	JSUSP	SUSP2	USBCKEN	—	EPS1	EPS0
AWR	AD6	AD5	AD4	AD3	AD2	AD1	AD0	WKEN
STL	—	—	—	—	STL3	STL2	STL1	STL0
SIES	NMI	UERR2	UERR1	UERR0	IN	OUT	UFERR	ASET
MISC	LEN0	READY	SETCMD	V33OS	—	CLEAR	TX	REQUEST
SETIO	—	—	—	—	SETIO3	SETIO2	SETIO1	DATATG
FRNUM0	D7	D6	D5	D4	D3	D2	D1	D0
FRNUM1	—	—	—	—	—	D10	D9	D8
FIFO0	D7	D6	D5	D4	D3	D2	D1	D0
FIFO1	D7	D6	D5	D4	D3	D2	D1	D0
FIFO2	D7	D6	D5	D4	D3	D2	D1	D0
FIFO3	D7	D6	D5	D4	D3	D2	D1	D0

USB Interface Register List

• SYSC Register

Bit	7	6	5	4	3	2	1	0
Name	BITWE	USBDIS	RUBUS	UBUSF	—	—	D1	ESDF
R/W	R/W	R/W	R/W	R	—	—	R/W	R/W
POR	0	0	0	1	—	—	0	x

"x": Unknown

Bit 7 **BITWE**: Software write permission enable control for certain bits
0: Enable – Software can write the URST, UFERR, LEN0 and SETCMD bits
1: Disable – Software can't write the URST, UFERR, LEN0 and SETCMD bits

Bit 6 **USBDIS**: USB SIE function control
0: Enable
1: Disable

This bit is used to control the USB SIE function. When this bit is set to 1, the USB SIE function will be disabled.

Bit 5 **RUBUS**: UBUS pin pull low function control
0: Enable
1: Disable

Bit 4 **UBUSF**: UBUS pin input status
0: Low level
1: High level

Bit 3~2 Unimplemented, read as "0"

Bit 1 **D1**: Reserved bit, cannot be used and must be fixed at 0

Bit 0 **ESDF**: ESD issue flag
This bit will be set to 1 when there is an ESD issue. It is set by SIE and cleared by software.

• USB_STAT Register

Bit	7	6	5	4	3	2	1	0
Name	OD1O	OD0O	OD1I	OD0I	SE1	SE0	PU	RUPH
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
POR	1	1	x	x	0	0	0	1

"x": Unknown

Bit 7 **OD1O**: Output data on OD1 pin, open drain NMOS output

Bit 6 **OD0O**: Output data on OD0 pin, open drain NMOS output

Bit 5 **OD1I**: OD1 pin input status

Bit 4 **OD0I**: OD0 pin input status

Bit 3 **SE1**: USB bus SE1 noise indication
This bit is used to indicate that the SIE has detected an SE1 noise on the USB bus. This bit is set by SIE and cleared by software.

Bit 2 **SE0**: USB bus SE0 noise indication
This bit is used to indicate that the SIE has detected an SE0 noise on the USB bus. This bit is set by SIE and cleared by software.

Bit 1 **PU**: UDP/UDN pins pull-high function control
0: Disable – no internal pull-high resistor
1: Enable – internal 600kΩ pull-high resistor on UDP/UDN pins

Bit 0 **PUPH**: 1.5kΩ resistor connection control
0: No 1.5kΩ resistor is connected between UDP and V33O
1: A 1.5kΩ resistor is connected between UDP and V33O

• UINT Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	EP3EN	EP2EN	EP1EN	EP0EN
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **EP3EN**: USB endpoint 3 interrupt enable control
0: Disable
1: Enable

Bit 2 **EP2EN**: USB endpoint 2 interrupt enable control
0: Disable
1: Enable

Bit 1 **EP1EN**: USB endpoint 1 interrupt enable control
0: Disable
1: Enable

Bit 0 **EP0EN**: USB endpoint 0 interrupt enable control
0: Disable
1: Enable

• **USC Register**

Bit	7	6	5	4	3	2	1	0
Name	URD	UMS2	UMS1	UMS0	RESUME	URST	RMWK	SUSP
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R
POR	1	0	0	0	x	x	x	x

"x": Unknown

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

Bit 6~4 **UMS2~UMS0**: USB and OD mode selection
000: No mode available – The V33O output will be floating. The relevant external pins will be in an input floating state.
001: Open drain output mode – The V33O output will be pulled high to VDD. The relevant external pins will become OD0/OD1 pins with a pull-high resistor respectively connected to VDD.
01x: USB mode – The V33O function will be enabled. The relevant external pins will be used as the UDP and UDN pins.
100~111: Undefined, the OD0/OD1 will be in a floating state.

Bit 3 **RESUME**: USB resume indication
0: Resume signal is not asserted or USB device has left the suspend mode
1: Resume signal is asserted and USB device is going to leave the suspend mode
This bit is read only. When the resume event occurs, this bit will be set high by SIE and then an interrupt will also be generated to wake up the MCU. In order to detect the suspend state, the MCU should set the USBCKEN bit to 1 and clear the SUSP2 bit to 0. When the USB device leaves the suspend mode, the SUSP bit will be cleared to 0 and then the RESUME bit will also be cleared to 0. The resume signal which causes the MCU to wake up should be noted and taken into consideration when the MCU is detecting the suspend mode.

Bit 2 **URST**: USB reset indication
0: No USB reset event occurs
1: USB reset event occurs
This bit is set by the SIE and only cleared by the MCU when the BITWE bit is cleared to zero. When the URST bit is set high, it indicates that a USB reset event has occurred and a USB interrupt will be generated.

Bit 1 **RMWK**: USB remote wake-up command
0: No USB remote wake-up command initiated
1: Initiate USB remote wake-up command
The RMWK bit is set to 1 by the MCU to force the USB host leaving the suspend mode. Setting the RMWK bit to 1 will initiate a remote wake-up command. The RMWK bit should be kept high for at least 1 USB clock to make sure that the remote wake-up command is accepted by the SIE.

Bit 0 **SUSP**: USB suspend indication
0: USB leaves the suspend mode
1: USB enters the suspend mode
This bit is read only and set to 1 by the SIE to indicate that the USB has entered the suspend mode. The corresponding interrupt will also be generated when the SUSP bit changes from low to high.

• UESR Register

The UESR register is the USB endpoint interrupt status register and is used to indicate which endpoint is accessed and to select the USB bus. The endpoint request flags, EPnF, are used to indicate which endpoints are accessed. If an endpoint is accessed, the related endpoint request flag will be set to "1" and the USB interrupt will occur if the USB interrupt is enabled and the stack is not full. When the active endpoint request flag is serviced, the endpoint request flag has to be cleared to "0" by software.

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	EP3F	EP2F	EP1F	EP0F
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 **EP3F**: Endpoint 3 access interrupt request flag
0: Not accessed
1: Accessed

Bit 2 **EP2F**: Endpoint 2 access interrupt request flag
0: Not accessed
1: Accessed

Bit 1 **EP1F**: Endpoint 1 access interrupt request flag
0: Not accessed
1: Accessed

Bit 0 **EP0F**: Endpoint 0 access interrupt request flag
0: Not accessed
1: Accessed

• UCC Register

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

"x": Unknown

Bit 7 **RCTRL**: R_{UDP2} resistor between UDP and UBUS connection control
0: Disable – No R_{UDP2} resistor is connected between UDP and UBUS lines
1: Enable – R_{UDP2} resistor is connected between UDP and UBUS lines

Bit 6 Unimplemented, read as "0"

Bit 5 **JSUSP**: USB J-STATE Suspend mode Indication
0: USB interface is not in the J-STATE suspend mode
1: USB interface is in the J-STATE suspend mode
This bit indicates whether the USB interface is in the J-STATE suspend mode or not.

Bit 4 **SUSP2**: USB suspend mode current reduction control
0: Current reduction is disabled in suspend mode
1: Current reduction is enabled in suspend mode
The current can be reduced to meet the USB standard specification if this bit is set to 1 when entering the suspend mode.

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

Bit 2 Unimplemented, read as "0"

Bit 1~0 **EPS1~EPS0**: Endpoint FIFO access selection
00: Endpoint 0 FIFO is selected
01: Endpoint 1 FIFO is selected
10: Endpoint 2 FIFO is selected
11: Endpoint 3 FIFO is selected

• AWR Register

The AWR register contains the current address and a remote wake up function control bit. The initial value of AWR is "00H". Whether the address value extracted from the USB host command is immediately loaded into this register or not is determined by the ASET bit in the SIES register.

Bit	7	6	5	4	3	2	1	0
Name	AD6	AD5	AD4	AD3	AD2	AD1	AD0	WKEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": Unknown

Bit 7~1 **AD6~AD0**: USB device address bit 6 ~ bit 0

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

• STL Register

The STL register shows whether the corresponding endpoint has worked properly or not. As soon as an endpoint improper operation occurs, the related bit in the STL register has to be set high by application program. The STL register content will be cleared by a USB reset signal and a setup token event.

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	STL3	STL2	STL1	STL0
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 **STL3**: USB endpoint 3 FIFO stall indication
0: Endpoint 3 FIFO is not stalled
1: Endpoint 3 FIFO is stalled

Bit 2 **STL2**: USB endpoint 2 FIFO stall indication
0: Endpoint 2 FIFO is not stalled
1: Endpoint 2 FIFO is stalled

Bit 1 **STL1**: USB endpoint 1 FIFO stall indication
0: Endpoint 1 FIFO is not stalled
1: Endpoint 1 FIFO is stalled

Bit 0 **STL0**: USB endpoint 0 FIFO stall indication
0: Endpoint 0 FIFO is not stalled
1: Endpoint 0 FIFO is stalled

• SIES Register

The SIES register is used to indicate the present signal state which the SIE receives and also controls whether the SIE changes the device address automatically or not.

Bit	7	6	5	4	3	2	1	0
Name	NMI	UERR2	UERR1	UERR0	IN	OUT	UFERR	ASET
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": Unknown

Bit 7 **NMI**: NAK token interrupt mask control
0: NAK token interrupt is not masked
1: NAK token interrupt is masked
If this bit is set to 1, the interrupt will not be generated when the device sends an NAK token to the USB host. Otherwise, the endpoint n NAK token interrupt will be generated if the corresponding endpoint interrupt control is enabled when this bit is cleared to 0 and the device sends an NAK token to the USB host.

Bit 6~4 **UERR2~UERR0**: USB SIE error status
0xx: No error
100: USB PID error
101: Bit stuffing error
110: CRC error
111: Host no response to SIE
These bits indicate which kind of error is detected by the USB SIE. These bits are set by the USB SIE and cleared by the application program.

Bit 3 **IN**: IN token indication
0: The received token packet is not IN token
1: The received token packet is IN token
The IN bit is used to indicate whether the current token packet received from the USB host is IN token or not.

Bit 2 **OUT**: OUT token indication
0: The received token packet is not OUT token
1: The received token packet is OUT token
The OUT bit is used to indicate whether the token received from the USB host is OUT token or not except the OUT zero length token. This bit should be cleared to 0 by application program after an OUT data has been read. Note that this bit will also be cleared when the next valid SETUP token is received.

Bit 1 **UFERR**: FIFO access error indication
0: No error occurs
1: Error occurs
This bit is used to indicate whether the USB bus errors, such as CRC error, PID error or bit stuffing error, etc., has occurred or not when the FIFO is accessed. This bit is set by SIE and cleared by application program when the BITWE bit is cleared to zero.

Bit 0 **ASET**: Device address update method control
0: Device address is immediately updated when an address is written into the AWR register
1: Device address is updated after the device IN token data have completely been read by the USB host
This bit is used to configure the SIE to automatically change the device address by the value stored in the AWR register. When this bit is set to "1" by firmware, the SIE will update the device address by the value stored in the AWR register after the USB host has successfully read the data from the device by an IN operation. Otherwise, when this bit is cleared to "0", the SIE will update the device address immediately after an address is written to the AWR register. Therefore, in order to operate properly, the firmware has to clear this bit after a next valid SETUP token is received.

• **MISC Register**

Bit	7	6	5	4	3	2	1	0
Name	LEN0	READY	SETCMD	V33OS	—	CLEAR	TX	REQUEST
R/W	R/W	R	R/W	R/W	—	R/W	R/W	R/W
POR	x	x	x	0	—	x	x	x

"x": Unknown

Bit 7 **LEN0**: Zero-length packet indication
0: The received packet is not zero-length packet
1: The received packet is zero-length packet
This bit is used to show whether the USB host sends a zero-length packet or not. It is set by Hardware and cleared by software when the BITWE bit is cleared to zero. It will also be cleared by hardware after the MCU receives a valid USB SETUP token.

Bit 6 **READY**: Endpoint FIFO ready indication
0: The desired endpoint FIFO is not ready
1: The desired endpoint FIFO is ready

Bit 5 **SETCMD**: SETUP command indication
0: The data in the FIFO is not SETUP token
1: The data in the FIFO is SETUP token
This bit is set by hardware and cleared by application program when the BITWE bit is cleared to zero.

Bit 4 **V33OS**: Voltage select
0: Internal V33O voltage
1: External 3.3V LDO

Bit 3 Unimplemented, read as "0"

Bit 2 **CLEAR**: FIFO clear function enable control
0: No operation
1: Clear the requested endpoint FIFO
This bit is used to clear the requested FIFO even if the corresponding FIFO is not ready. The CLEAR bit should be set to 1 to generate a positive pulse with a pulse width to clear the requested FIFO and then clear this bit to zero. After clearing the FIFO, the USB interface Out pipe endpoint can receive new data from the Host and In pipe endpoint can transfer new data to the Host.

Bit 1 **TX**: Data transfer direction indication
0: MCU read data from the USB FIFO
1: MCU wirte data to the USB FIFO
This bit defines the data transfer direction between the MCU and USB endpoint FIFO. When the TX bit is set to 1, it means that the MCU wants to write data to the USB endpoint FIFO. After the MCU write operation has completed, this bit has to be cleared to 0 before terminating the FIFO request to indicate the end of the data transfer. For an MCU read operation this bit has to be cleared to 0 to indicate that the MCU wants to read data from the USB endpoint FIFO. Then this bit has to be set to 1 before terminating the FIFO request to indicate the end of the data transfer after an MCU read operation completion.

Bit 0 **REQUEST**: FIFO request control
0: No request or request completion
1: Request desired FIFO
This bit is used to request an operation of the desired endpoint FIFO. After selecting the desired endpoint FIFO, the FIFO can be requested by setting this bit high. Then this bit should be cleared to zero after the operation completion.

• **SETIO Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SETIO3	SETIO2	SETIO1	DATATG
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **SETIO3**: Endpoint 3 FIFO control
0: Output pipe
1: Input pipe

Bit 2 **SETIO2**: Endpoint 2 FIFO control
0: Output pipe
1: Input pipe

Bit 1 **SETIO1**: Endpoint 1 FIFO control
0: Output pipe
1: Input pipe

Bit 0 **DATATG**: Data token toggle bit
0: DATA0 will be sent first
1: DATA1 will be sent first
This bit is used to select the Data token toggle bit. When this bit is cleared to 0, a DATA0 will be sent first in the following IN or OUT data pipe for the requested endpoint FIFO. Otherwise, a DATA1 will be sent first followed by the successive IN or OUT data transfer.

• **FRNUM0 Register**

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

"x": Unknown

Bit 7~0 **D7~D0**: Frame number D7~D0 for each USB SOF package

• **FRNUM1 Register**

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

"x": Unknown

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **D10~D8**: Frame number D10~D8 for each USB SOF package

• **FIFOn Registers**

The FIFOn Register is used for data transactions storages between the USB device and the USB host. The MCU reads data from or writes data to the FIFOn via the specific combination of the corresponding control and selection bits.

Name	Type	POR	Descriptions
FIFO0	R/W	xxxx xxxx	Endpoint 0 Data Pipe
FIFO1	R/W	xxxx xxxx	Endpoint 1 Data Pipe
FIFO2	R/W	xxxx xxxx	Endpoint 2 Data Pipe
FIFO3	R/W	xxxx xxxx	Endpoint 3 Data Pipe

"x": Unknown

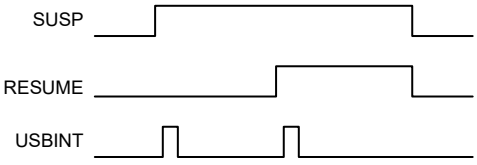
USB Suspend Mode and Wake-Up

USB Suspend Mode

If there is no signal on the USB bus for over 3ms, the USB device will enter the suspend mode. The Suspend flag, SUSP, in the USC register will then be set high and an USB interrupt will be generated to indicate that the device should jump to the suspend state to meet the requirements of the USB suspend current specification. In order to meet the requirements of the suspend current; the firmware should disable the USB clock by clearing the USBCKEN bit to "0". The suspend mode current can be further decreased by setting the SUSP2 bit in the UCC register.

USB Host Wake-up

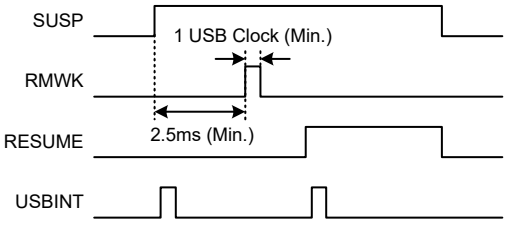
When the resume signal is asserted by the USB host, the device will be woken up by the USB interrupt and the RESUME bit in the USC register will be set. To ensure correct device operation, the application program should set the USBCKEN bit high and the USB host will start to communicate with the USB device. Then the SUSP bit will be cleared low together with the RESUME bit when the USB device actually leaves the suspend mode. Therefore, when the device detects the suspend bit, SUSP, the resume bit, RESUME, should be monitored and taken into consideration.



Suspend and Host Wake-up

USB Remote Wake-up

As the USB devic has a remote wake-up function, the USB device can wake up the USB host by sending a remote wake-up pulse which is generated by setting the RMWK bit high. Once the USB host receives a remote wake-up signal from the USB device, the host will send a resume signal to the device.



Suspend and Remote Wake-up

USB Interrupts

Several USB conditions can generate a USB interrupt. When one of these conditions exists, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are the USB suspended, USB resumed, USB reset and USB endpoint FIFO access events. When the USB interrupt caused by any of these conditions occurs, if the corresponding interrupt control is enabled and the stack is not full, the program will jump to the corresponding interrupt vector where it can be serviced before returning to the main program.

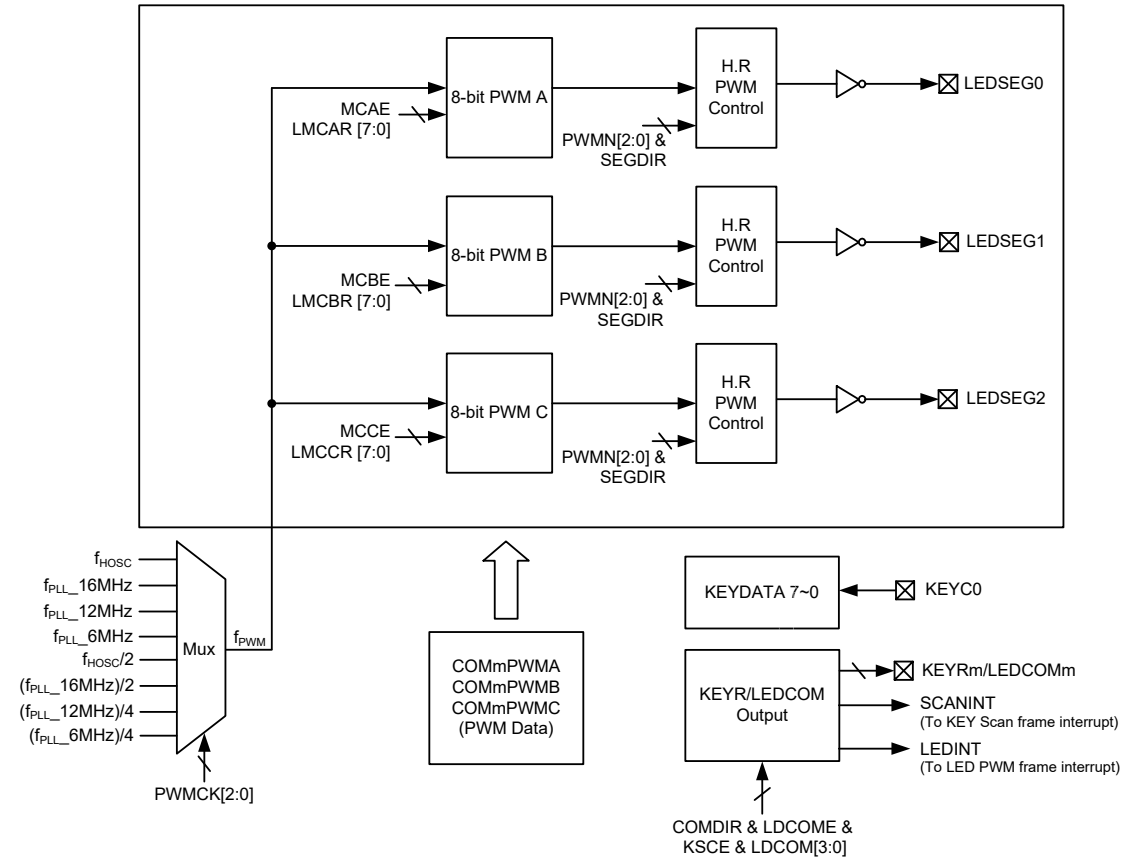
For the USB Endpoint FIFO access event, there are the corresponding indication flags to indicate which endpoint FIFO is accessed. As the Endpoint FIFO access flag is set, it will generate a USB interrupt if the associated Endpoint FIFO pipe and interrupt control are both enabled. The Endpoint

FIFO access flags should be cleared by the application program. As the USB suspended or USB resume condition occurs, the corresponding indication flag, known as SUSP and RESUME bits, will be set and a USB interrupt will directly generate without enabling the associated interrupt control bit. The SUSP and RESUME bits are read only and set or cleared by the USB SIE. For a USB interrupt occurred to be serviced, in addition to the bits for the corresponding interrupt enable control in USB module being set, the global interrupt enable control and the related interrupt enable control bits in the host MCU must also be set. If these bits are not set, then no interrupt will be serviced.

In addition, each USB start of frame package will generate an interrupt if the global interrupt enable control and the related interrupt enable control bit SOFE are both set. Refer to the Interrupt section for more detailed information.

LED PWM Function – HT68FB541

The HT68FB541 device includes a PWM function for LED application, which is composed of an LED PWM module, an LED COM output unit and several registers.



LED PWM Block Diagram (m=0~7)

LED PWM Registers

Overall operation of the LED PWM function is controlled using a series of registers.

Register Name	Bit							
	7	6	5	4	3	2	1	0
LMCE	—	—	—	—	—	MCCE	MCBE	MCAE
LMCAR	MCAD7	MCAD6	MCAD5	MCAD4	MCAD3	MCAD2	MCAD1	MCAD0
LMCBR	MCBD7	MCBD6	MCBD5	MCBD4	MCBD3	MCBD2	M0CBD1	MCBD0
LMCCR	MCCD7	MCCD6	MCCD5	MCCD4	MCCD3	MCCD2	M1CAD1	MCCD0
PWMCTL0	PWMN2	PWMN1	PWMN0	PWMCK2	PWMCK1	PWMCK0	—	LDCOME
PWMCTL1	KEYSR4	KEYSR3	KEYSR2	KEYSR1	KEYSR0	PWMGE	—	KSCE
PWMCTL2	TDELAY3	TDELAY2	TDELAY1	TDELAY0	—	—	COMDIR	SEGDIR
PWMCTL3	COMRD3	COMRD2	COMRD1	COMRD0	LDCOM3	LDCOM2	LDCOM1	LDCOM0
PWMCmA	D7	D6	D5	D4	D3	D2	D1	D0
PWMCmB	D7	D6	D5	D4	D3	D2	D1	D0
PWMCmC	D7	D6	D5	D4	D3	D2	D1	D0
KEYDATAx	—	—	—	—	—	—	—	D0

LED PWM Register List (x=0~7, m=0~7)

• LMCE Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	MCCE	MCBE	MCAE
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **MCCE**: Module C PWM function enable control
0: Disable, C PWM counter=0
1: Enable

Bit 1 **MCBE**: Module B PWM function enable control
0: Disable, B PWM counter=0
1: Enable

Bit 0 **MCAE**: Module A PWM function enable control
0: Disable, A PWM counter=0
1: Enable

Note: This register can be read/written regardless of the LDCOME bit state. When the PWMGE bit is zero, the LEDSEG is inactive regardless of the LDCOME bit state. When the PWMGE=1 and LDCOME=0, the PWM counter will be enabled if the MCAE/MCBE/MCCE bit is set high, otherwise the LEDSEG will be inactive if the MCAE/MCBE/MCCE bit is zero. When PWMGE=1 and LDCOME=1, the PWM counter cannot be controlled regardless of the MCAE/MCBE/MCCE bit state.

• LMCAR Register

Bit	7	6	5	4	3	2	1	0
Name	MCAD7	MCAD6	MCAD5	MCAD4	MCAD3	MCAD2	MCAD1	MCAD0
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 **MCAD7~MCAD0**: Duty data for Module 8-bit A PWM

This register can be read or written if the LDCOME bit is cleared to 0 to select the manual mode. When the LDCOME bit is set, the register can only be read. An excessive change value may be read from this register during the PWM data transferring period.

• **LMCBR Register**

Bit	7	6	5	4	3	2	1	0
Name	MCBD7	MCBD6	MCBD5	MCBD4	MCBD3	MCBD2	M0CBD1	MCBD0
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 **MCBD7~MCBD0**: Duty data for Module 8-bit B PWM

This register can be read or written if the LDCOME bit is cleared to 0 to select the manual mode. When the LDCOME bit is set, the register can only be read. An excessive change value may be read from this register during the PWM data transferring period.

• **LMCCR Register**

Bit	7	6	5	4	3	2	1	0
Name	MCCD7	MCCD6	MCCD5	MCCD4	MCCD3	MCCD2	M1CAD1	MCCD0
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 **MCCD7~MCCD0**: Duty data for Module 8-bit C PWM

This register can be read or written if the LDCOME bit is cleared to 0 to select the manual mode. When the LDCOME bit is set, the register can only be read. An excessive change value may be read from this register during the PWM data transferring period.

• **PWMCTL0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PWMN2	PWMN1	PWMN0	PWMCK2	PWMCK1	PWMCK0	—	LDCOME
R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	R/W
POR	0	0	0	0	0	0	—	0

Bit 7~5 **PWMN2~PWMN0**: Number of PWM waveform output on each LEDCOM
000: 8 times
001: 7 times
010: 6 times
011: 5 times
100: 4 times
101: 3 times
110: 2 times
111: 1 time
These bits can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

Bit 4~2 **PWMCK2~PWMCK0**: LED PWM module clock source selection (f_{PWM})
000: f_{HOSC}
001: f_{PLL}_16MHz
010: f_{PLL}_12MHz
011: f_{PLL}_6MHz
100: $f_{HOSC}/2$
101: (f_{PLL}_16MHz)/2
110: (f_{PLL}_12MHz)/4
111: (f_{PLL}_6MHz)/4
These bits can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

Bit 1 Unimplemented, read as “0”

Bit 0 **LDCOME**: LED hardware COM function enable control
0: Disable, manual mode, users should fill in the related registers by the application program
1: Enable, auto mode, the LED hardware automatically completes the related duty, mode operation and COM output
This bit can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

• PWMCTL1 Register

Bit	7	6	5	4	3	2	1	0
Name	KEYSR4	KEYSR3	KEYSR2	KEYSR1	KEYSR0	PWMGE	—	KSCE
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 7~3 **KEYSR4~KEYSR0**: Hardware key scan output function
00000: KEYR0 active
00001: KEYR0~1 active
00010: KEYR0~2 active
00011: KEYR0~3 active
00100: KEYR0~4 active
00101: KEYR0~5 active
00110: KEYR0~6 active
00111~11111: KEYR0~7 active
These bits can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

Bit 2 **PWMGE**: Global LED hardware function enable control
0: Disable
1: Enable
When this bit is zero, the LED PWM function will be disabled, all the LEDSEG and LEDCOM outputs will be in an inactive state, the PWM clock will be blocked. When this bit changes from high to low, the PWM function will not be disabled until the current LED frame and Scan frame are both completed. If the PWMGE bit changes from high to low and then is set to high again, such transition which occurs during an LED frame or Key Scan frame duration will not disable the PWM function, instead it will remain in the enable state.

Bit 1 Unimplemented, read as "0"

Bit 0 **KSCE**: Hardware key scan function enable control
0: Disable
1: Enable
This bit can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

• PWMCTL2 Register

Bit	7	6	5	4	3	2	1	0
Name	TDELAY3	TDELAY2	TDELAY1	TDELAY0	—	—	COMDIR	SEGDIR
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7~4 **TDELAY3~TDELAY0**: Hardware key scan delay time selection
0000: $5/f_{PWM}$
0001: $10/f_{PWM}$
0010: $20/f_{PWM}$
0011: $40/f_{PWM}$
0100: $80/f_{PWM}$
0101: $160/f_{PWM}$
0110: $320/f_{PWM}$

0111: 640/f_{PWM}
1000: 4/f_{PWM}
1001: 8/f_{PWM}
1010: 16/f_{PWM}
1011: 32/f_{PWM}
1100: 64/f_{PWM}
1101: 128/f_{PWM}
1110: 256/f_{PWM}
1111: 512/f_{PWM}

These bits can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **COMDIR**: LED COM output polarity
0: High active
1: Low active

This bit can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

Bit 0 **SEGDIR**: LED SEG output polarity
0: Low active
1: High active

This bit can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

• PWMCTL3 Register

Bit	7	6	5	4	3	2	1	0
Name	COMRD3	COMRD2	COMRD1	COMRD0	LDCOM3	LDCOM2	LDCOM1	LDCOM0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~4 **COMRD3~COMRD0**: Indicate which LEDCOM is being processed by hardware
0000: LEDCOM0
0001: LEDCOM1
0010: LEDCOM2
0011: LEDCOM3
0100: LEDCOM4
0101: LEDCOM5
0110: LEDCOM6
0111: LEDCOM7
1000~1111: Reserved

These bits are read only and are used to indicate that which LEDCOM is being processed by Hardware. If the LED PWM function is disabled, this bit field value has no effect.

Bit 3~0 **LDCOM3~LDCOM0**: Hardware LED COM output selection
0000: LEDCOM0 output
0001: LEDCOM0~1 output
0010: LEDCOM0~2 output
0011: LEDCOM0~3 output
0100: LEDCOM0~4 output
0101: LEDCOM0~5 output
0110: LEDCOM0~6 output
0111~1111: LEDCOM0~7 output

Note: The Hardware always chooses the selected LEDCOM(s) to output according to the data in LDCOM3~LDCOM0. But in the condition of m < Hardware LED COM output selection value, the hardware will keep the corresponding PWM waveform output disabled in the extra LEDCOM(s).

These bits can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

• **KEYDATAx Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **D0**: Hardware key scan input pin KEYC0 input data 0 for scan output LEDCOMm

• **PWMCmA Register**

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

Bit 7~0 **D7~D0**: PWM duty Data of PWMA for LEDCOMm in the Auto mode

• **PWMCmB Register**

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

Bit 7~0 **D7~D0**: PWM duty Data of PWMB for LEDCOMm in the Auto mode

• **PWMCmC Register**

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

Bit 7~0 **D7~D0**: PWM duty Data of PWMC for LEDCOMm in the Auto mode

LED PWM Modules

The HT68FB541 device provides an LED PWM module, which contains three 8-bit PWMs, namely PWMA, PWMB and PWMC. The LED PWM module has three segment outputs.

LED COM Output Unit

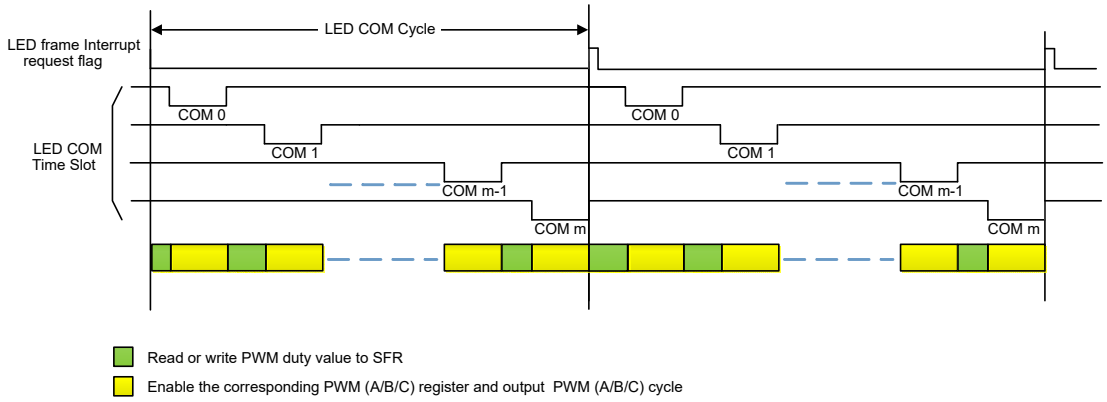
The HT68FB541 device provides an LED COM output unit used to output external LEDCOMm enable control signal. By using the Time Shared method combined with the LED PWM module, 24 LED Segment PWM outputs can be generated, which can implement up to 24 LED applications. The number of LED COM output is determined by the LDCOM[3:0] bits, by which 1 to 8 LEDCOM outputs can be selected.

LED Operation

The LED function can be configured to operate in manual mode or auto mode, which is determined by the LDCOME bit. If the LDCOME bit is set high to select the auto mode, the LED function will automatically complete the related PWM duty and LED COM output by hardware. The HT68FB541 device provides several Auto mode PWM duty data registers, namely the PWMCmA, PWMCmB and PWMCmC registers, to record the LED PWM duty data, these data will be converted to the related registers by the hardware and then generates 24 LED segment PWM signals. If the LDCOME bit is cleared to zero, users can also control the PWM output in manual mode by using

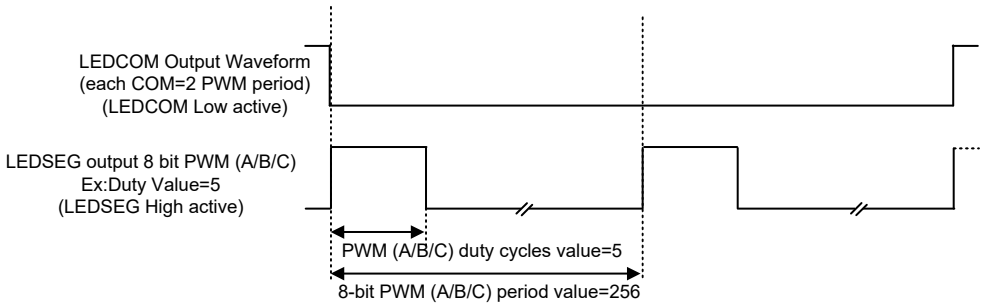
the LED PWM module, PWM register enable signals and duty data. The COMDIR and SEGDIR bits are used to setup the LEDCOM and LEDSEG output polarity, special attention should be paid to this point in application programs.

The timing of the hardware LED PWM module, i.e. 3 LED Segment PWM and 8 LED COMs, is shown as follow, where the LEDCOM is low active and LEDSEG is high active:



LED PWM Timing Diagram (m=0~7)

The PWM Module PWM output waveform is as follows:



Module PWM Output Waveform

Hardware Auto Key Scan Function

The LED PWM also supports a Hardware Auto Key Scan function, which can be enabled by setting the KSCE bit. This function allows the data on the key scan output ports KEYRm, the ports which are pin-shared with LEDCOMm functions, to be input through the key scan input pin KEYC0 during the LED COM output or Key Scan output duration, the input data on the KEYC0 pin will then be captured and stored into the KEYDATAx registers at the instant the LED COM output changed to an inactive state. This function can be configured using the related register bits, the KSCE, TDELAY3~TDELY0 and KEYSR4~KEYSR0 bits, as well as the KEYDATAx registers used for scan input data storage.

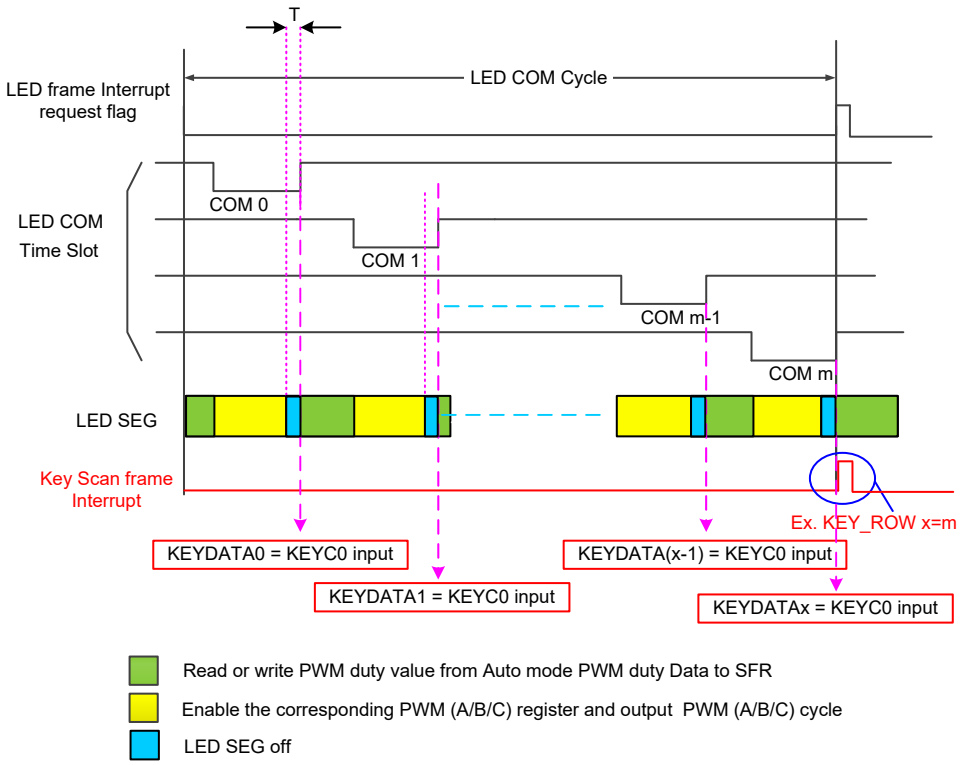
The related timings of LED COM output and Key Scan are shown in the following diagrams:

Condition (1):

KSCE=1; Enable hardware Key Scan function;

T1=TDELAY3~0;

KEYSR4~0 (x)=LDCOM3~0 (m);



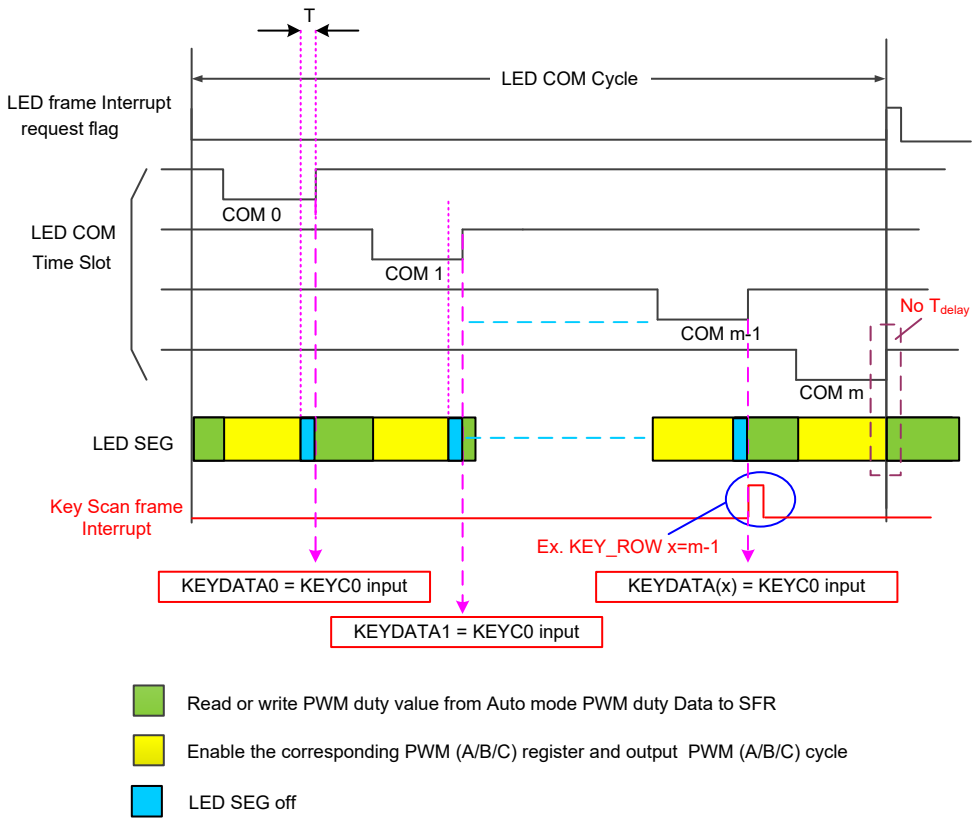
Condition (2):

KSCE=1; Enable hardware Key Scan function;

KEYSR4~0 (x) < LDCOM3~0 (m);

T1=TDELAY3~0;

Ex. x=m-1;



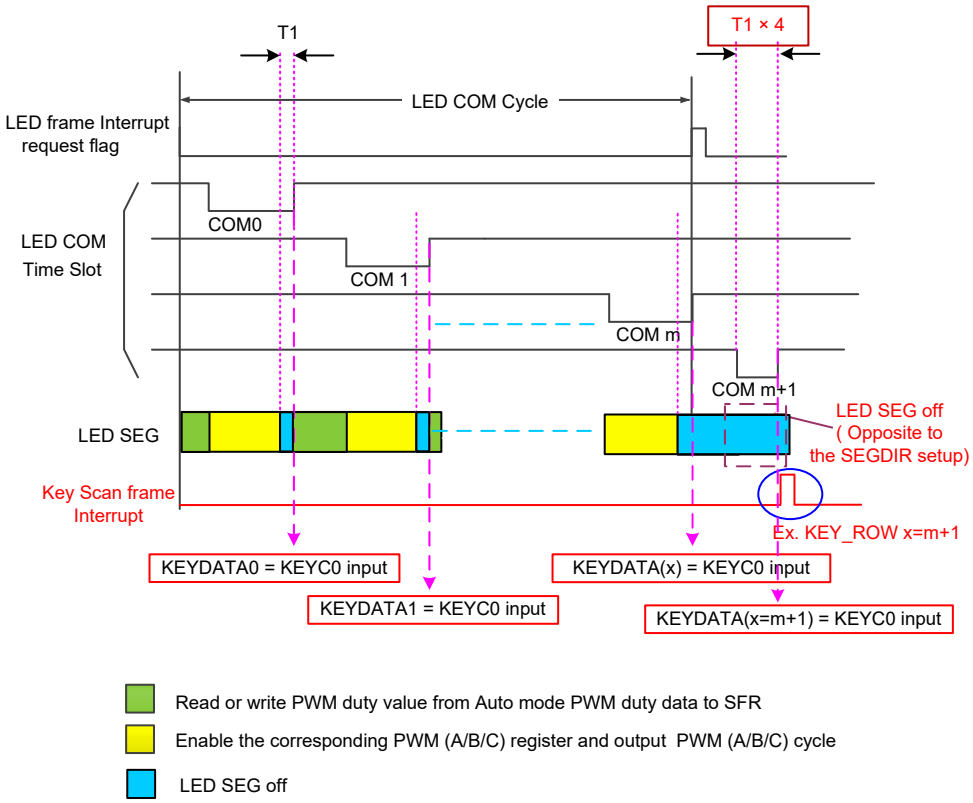
Condition (3):

KSCE=1; Enable hardware Key Scan function;

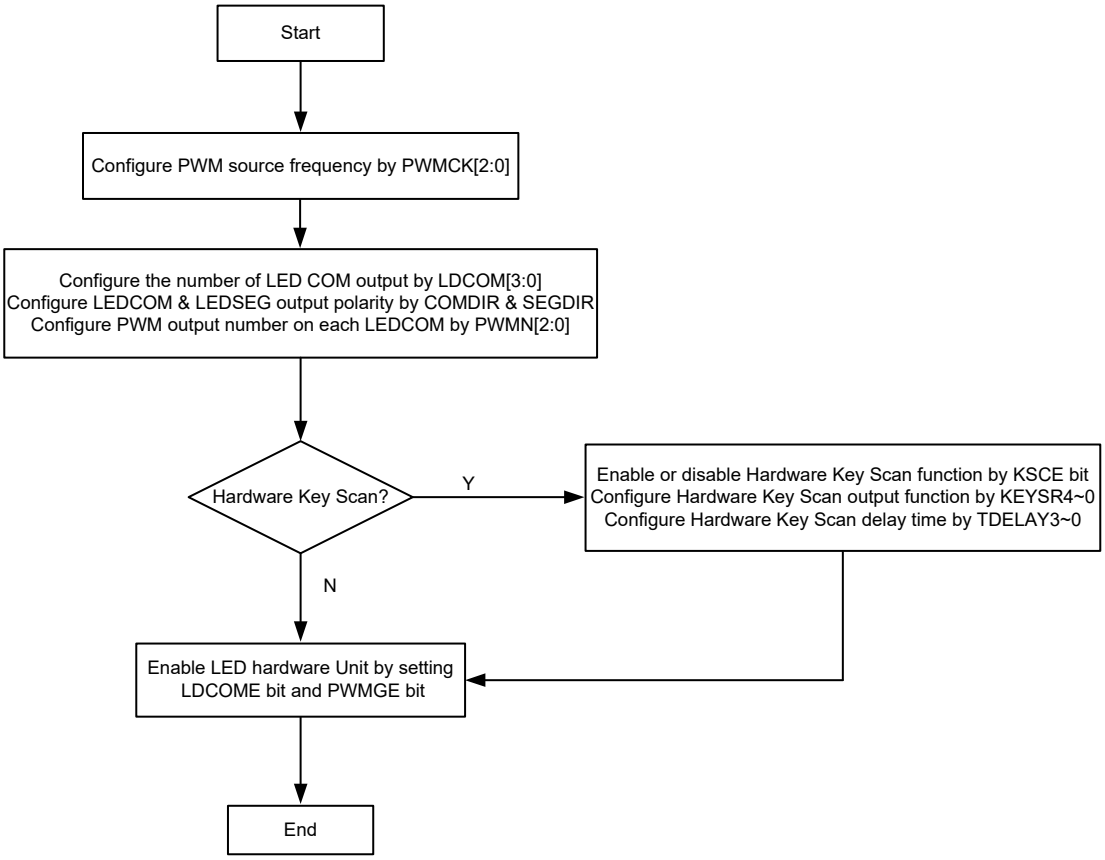
KEYSR4~0 (x) > LDCOM3~0 (m);

T1=TDELAY3~0;

Ex. x=m+1;



The global LED PWM function configuration flowchart is shown as below:



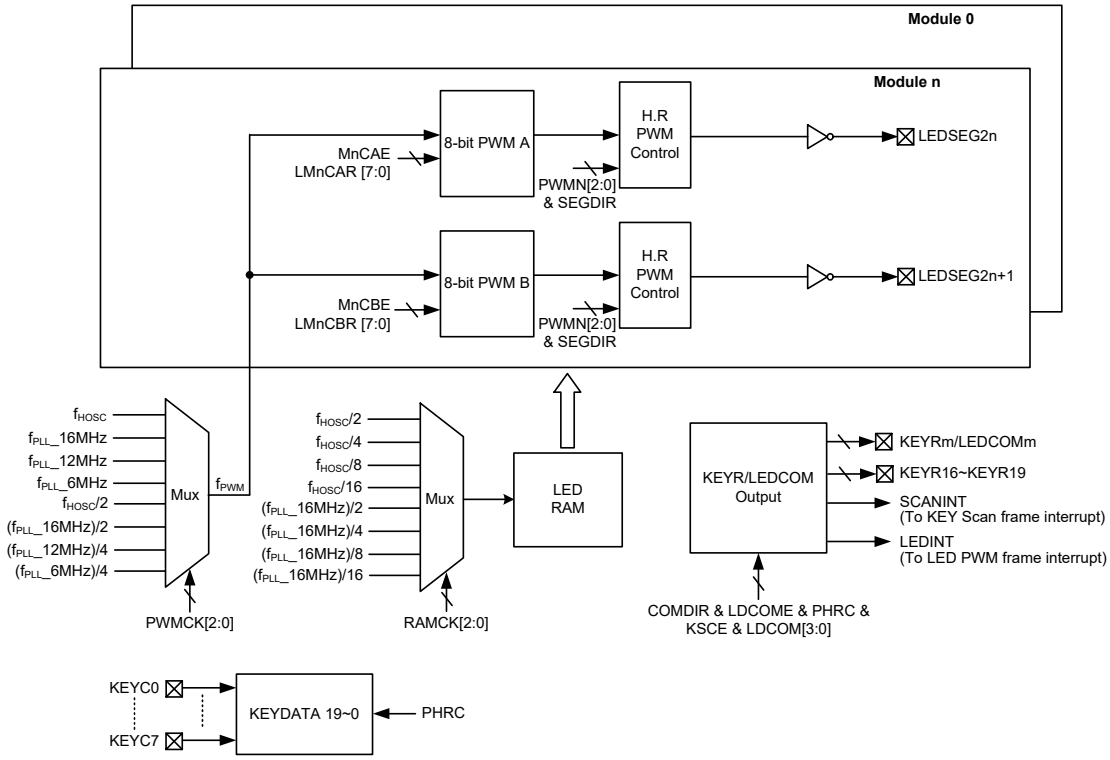
PWM Output Control Flow Chart

LED Interrupt

The LED function has two interrupt output signals, LED PWM frame interrupt and Key Scan frame interrupt. After one frame, i.e. 24 LED segment PWM signals, is completed by the hardware, an LED PWM frame interrupt request will be generated to inform the MCU. It should be noted that when the LED PWM is disabled by application program, it will not be disabled until the hardware completes the current PWM frame and sends out an interrupt request. When the Hardware Key Scan function is enabled, a Key Scan frame interrupt request will be generated to inform the MCU when the Key Scan frame is completed. Refer to the Interrupt section for more detailed information.

LED PWM Function – HT68FB571

The HT68FB571 device includes a PWM function for LED application, which is composed of four LED PWM modules, an LED RAM unit, an LED COM output unit and several registers.



LED PWM Block Diagram (n=0~3, m=0~15)

LED PWM Registers

Overall operation of the LED PWM function is controlled using a series of registers.

Register Name	Bit							
	7	6	5	4	3	2	1	0
LMnCE	—	—	—	—	—	—	MnCBE	MnCAE
LMnCAR	MnCAD7	MnCAD6	MnCAD5	MnCAD4	MnCAD3	MnCAD2	MnCAD1	MnCAD0
LMnCBR	MnCBD7	MnCBD6	MnCBD5	MnCBD4	MnCBD3	MnCBD2	MnCBD1	MnCBD0
PWMCTL0	PWMN2	PWMN1	PWMN0	PWMCK2	PWMCK1	PWMCK0	—	LDCOME
PWMCTL1	KEYSR4	KEYSR3	KEYSR2	KEYSR1	KEYSR0	PWMGE	PHRC	KSCE
PWMCTL2	TDELAY3	TDELAY2	TDELAY1	TDELAY0	TPHRE1	TPHRE0	COMDIR	SEGDIR
PWMCTL3	COMRD3	COMRD2	COMRD1	COMRD0	LDCOM3	LDCOM2	LDCOM1	LDCOM0
PWMCTL4	—	—	—	—	—	RAMCK2	RAMCK1	RAMCK0
KEYDATAx	D7	D6	D5	D4	D3	D2	D1	D0

LED PWM Register List (n=0~3, x=0~19)

• **LMnCE Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1 **MnCBE**: Module n B PWM function enable control
0: Disable, B PWM counter=0
1: Enable

Bit 0 **MnCAE**: Module n A PWM function enable control
0: Disable, A PWM counter=0
1: Enable

Note: This register can be read/written regardless of the LDCOME bit state. When the PWMGE bit is zero, the LEDSEG is inactive regardless of the LDCOME bit state. When the PWMGE=1 and LDCOME=0, the PWM counter will be enabled if the MnCAE/MnCBE bit is set high, otherwise the LEDSEG will be inactive if the MnCAE/MnCBE bit is zero. When PWMGE=1 and LDCOME=1, the PWM counter cannot be controlled regardless of the MnCAE/MnCBE bit state.

• **LMnCAR Register**

Bit	7	6	5	4	3	2	1	0
Name	MnCAD7	MnCAD6	MnCAD5	MnCAD4	MnCAD3	MnCAD2	MnCAD1	MnCAD0
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 **MnCAD7~MnCAD0**: Duty data for module n 8-bit A PWM

This register can be read or written if the LDCOME bit is cleared to 0 to select the manual mode. When the LDCOME bit is set, the register can only be read. An excessive change value may be read from this register during the RAM data transferring period.

• **LMnCBR Register**

Bit	7	6	5	4	3	2	1	0
Name	MnCBD7	MnCBD6	MnCBD5	MnCBD4	MnCBD3	MnCBD2	MnCBD1	MnCBD0
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 **MnCBD7~MnCBD0**: Duty data for module n 8-bit B PWM

This register can be read or written if the LDCOME bit is cleared to 0 to select the manual mode. When the LDCOME bit is set, the register can only be read. An excessive change value may be read from this register during the RAM data transferring period.

• **PWMCTL0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PWMN2	PWMN1	PWMN0	PWMCK2	PWMCK1	PWMCK0	—	LDCOME
R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	R/W
POR	0	0	0	0	0	0	—	0

Bit 7~5 **PWMN2~PWMN0**: Number of PWM waveform output on each LEDCOM
000: 8 times
001: 7 times

010: 6 times
011: 5 times
100: 4 times
101: 3 times
110: 2 times
111: 1 time

These bits can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

Bit 4~2 **PWMCK2~PWMCK0**: LED PWM modules clock source selection (f_{PWM})
000: f_{HOSC}
001: f_{PLL}_16MHz
010: f_{PLL}_12MHz
011: f_{PLL}_6MHz
100: f_{HOSC}/2
101: (f_{PLL}_16MHz)/2
110: (f_{PLL}_12MHz)/4
111: (f_{PLL}_6MHz)/4

These bits can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

Bit 1 Unimplemented, read as "0"

Bit 0 **LDCOME**: LED hardware COM function enable control
0: Disable, manual mode, users should fill in the related registers by the application program
1: Enable, auto mode, the LED hardware automatically completes the related duty, mode operation and COM output

This bit can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

- **PWMCTL1 Register**

Bit	7	6	5	4	3	2	1	0
Name	KEYSR4	KEYSR3	KEYSR2	KEYSR1	KEYSR0	PWMGE	PHRC	KSCE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~3 **KEYSR4~KEYSR0**: Hardware key scan output function
00000: KEYR 0 active
00001: KEYR 0~1 active
00010: KEYR 0~2 active
00011: KEYR 0~3 active
00100: KEYR 0~4 active
00101: KEYR 0~5 active
00110: KEYR 0~6 active
00111: KEYR 0~7 active
01000: KEYR 0~8 active
01001: KEYR 0~9 active
01010: KEYR 0~10 active
01011: KEYR 0~11 active
01100: KEYR 0~12 active
01101: KEYR 0~13 active
01110: KEYR 0~14 active
01111: KEYR 0~15 active
10000: KEYR 0~16 ative
10001: KEYR 0~17 active
10010: KEYR 0~18 active
10011~11111: KEYR 0~19 active

These bits can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

Bit 2 **PWMGE**: Global LED hardware function enable control
0: Disable
1: Enable
When this bit is zero, the LED PWM function will be disabled, all the LEDSEG and LEDCOM outputs will be in an inactive state, the PWM clock and RAM clock will be blocked. When this bit changes from high to low, the PWM function will not be disabled until the current LED frame and Scan frame are both completed. If the PWMGE bit changes from high to low and then is set to high again, such transition which occurs during an LED frame or Key Scan frame duration will not disable the PWM function, instead it will remain in the enable state.

Bit 1 **PHRC**: Hardware key scan input port pull high resistor control
0: Disable
1: Enable
This bit can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

Bit 0 **KSCE**: Hardware key scan function enable control
0: Disable
1: Enable
This bit can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

- **PWMCTL2 Register**

Bit	7	6	5	4	3	2	1	0
Name	TDELAY3	TDELAY2	TDELAY1	TDELAY0	TPHRE1	TPHRE0	COMDIR	SEGDIR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~4 **TDELAY3~TDELAY0**: Hardware key scan delay time selection
0000: 5/f_{PWM}
0001: 10/f_{PWM}
0010: 20/f_{PWM}
0011: 40/f_{PWM}
0100: 80/f_{PWM}
0101: 160/f_{PWM}
0110: 320/f_{PWM}
0111: 640/f_{PWM}
1000: 4/f_{PWM}
1001: 8/f_{PWM}
1010: 16/f_{PWM}
1011: 32/f_{PWM}
1100: 64/f_{PWM}
1101: 128/f_{PWM}
1110: 256/f_{PWM}
1111: 512/f_{PWM}
These bits can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

Bit 3~2 **TPHRE1~TPHRE0**: Time for hardware key scan input port pull high resistor
00: The last quarter of the last LED COM PWM cycle
01: 1/2 of the last LED COM PWM cycle
10: The last 3/4 of the last LED COM PWM cycle
11: The beginning of the first LED COM PWM cycle

Bit 1 **COMDIR**: LED COM output polarity
0: Low active
1: High active
This bit can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

Bit 0 **SEGDIR**: LED SEG output polarity
0: High active
1: Low active
This bit can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

• PWMCTL3 Register

Bit	7	6	5	4	3	2	1	0
Name	COMRD3	COMRD2	COMRD1	COMRD0	LDCOM3	LDCOM2	LDCOM1	LDCOM0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~4 **COMRD3~COMRD0**: Indicate which LEDCOM is being processed by Hardware
0000: LEDCOM0
0001: LEDCOM1
0010: LEDCOM2
0011: LEDCOM3
0100: LEDCOM4
0101: LEDCOM5
0110: LEDCOM6
0111: LEDCOM7
1000: LEDCOM8
1001: LEDCOM9
1010: LEDCOM10
1011: LEDCOM11
1100: LEDCOM12
1101: LEDCOM13
1110: LEDCOM14
1111: LEDCOM15
These bits are read only and are used to indicate that which LEDCOM is being processed by Hardware. If the LED PWM function is disabled, this bit field value has no effect.

Bit 3~0 **LDCOM3~LDCOM0**: Hardware LED COM output selection
0000: LEDCOM0 output
0001: LEDCOM0~1 output
0010: LEDCOM0~2 output
0011: LEDCOM0~3 output
0100: LEDCOM0~4 output
0101: LEDCOM0~5 output
0110: LEDCOM0~6 output
0111: LEDCOM0~7 output
1000: LEDCOM0~8 output
1001: LEDCOM0~9 output
1010: LEDCOM0~10 output
1011: LEDCOM0~11 output
1100: LEDCOM0~12 output
1101: LEDCOM0~13 output
1110: LEDCOM0~14 output
1111: LEDCOM0~15 output
Note: The Hardware always chooses the selected LEDCOM(s) to output according to the data in LDCOM3~LDCOM0. But in the condition of m < Hardware LED COM output selection value, the hardware will keep the corresponding PWM waveform output disabled in the extra LEDCOM(s).
These bits can only be changed when the LED PWM function is configured as manual mode or the PWMGE bit is disabled.

• **PWMCTL4 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	RAMCK2	RAMCK1	RAMCK0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	1	0

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **RAMCK2~RAMCK0**: LED RAM clock source selection
000: f_{HOSC}/2
001: f_{HOSC}/4
010: f_{HOSC}/8
011: f_{HOSC}/16
100: (f_{PLL}_16MHz)/2
101: (f_{PLL}_16MHz)/4
110: (f_{PLL}_16MHz)/8
111: (f_{PLL}_16MHz)/16

• **KEYDATAx Register**

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

Bit 7~0 **D7~D0**: Hardware key scan input port KEYC input data 7~0 for scan output LEDCOMm

LED PWM Modules

The HT68FB571 device provides four LED PWM modules, each module contains two 8-bit PWMs, namely PWMA and PWMB. Each LED PWM module has the identical architecture, the number of PWM segment outputs is 2(n+1) decided by the selected Module number n.

LED COM Output Unit

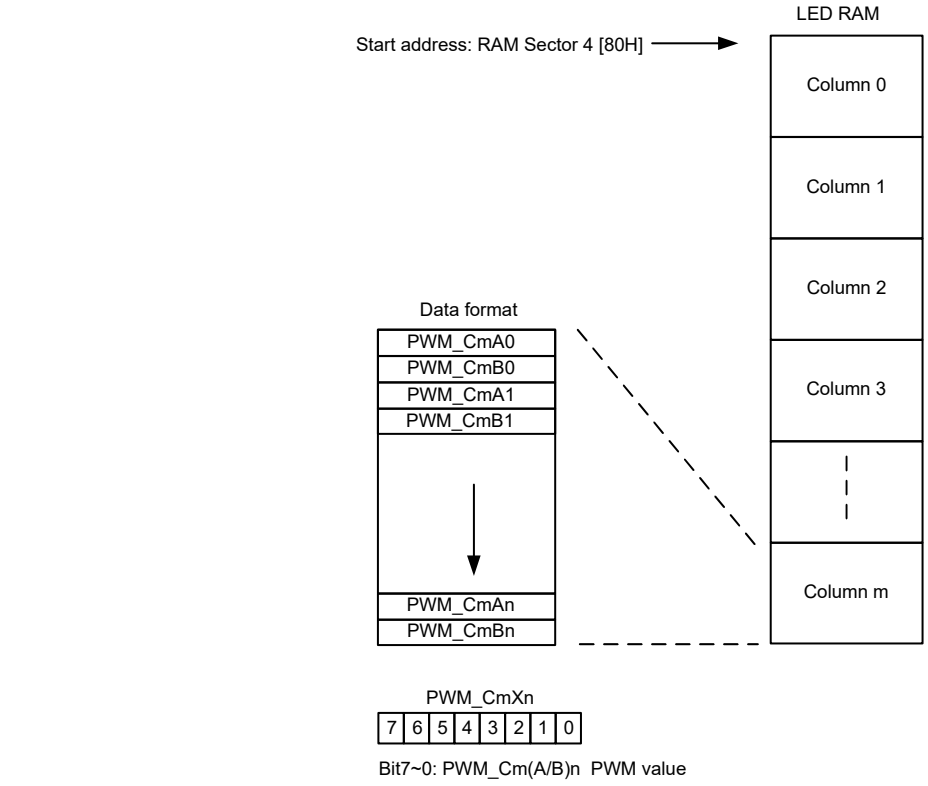
The HT68FB571 device provides an LED COM output unit used to output external LEDCOMm enable control signal. By using the Time Shared method combined with LED PWM modules, 128 LED Segment PWM outputs can be generated, which can implement up to 128 LED application. The number of LED COM output is determined by the LDCOM[3:0] bits, by which 1 to 16 LEDCOM outputs can be selected.

LED Memory

The HT68FB571 device provides an area of embedded data memory for the LED. This data area is known as the LED Memory, which is used to record LED PWM duty data. The LED RAM is used to convert the LED RAM data to the LED PWM module related registers data, and generates 128 LED segment PWM signals.

The LED Memory has a capacity of 128 bytes. The LED RAM is subdivided into 16 columns, namely Column 0~Column 15, corresponding respectively to LEDCOM0~LEDCOM15. Each column has 8 bytes data, which is recorded as PWM value with a specific order of PWM_A0 →PWM_B0→……PWM_A3→PWMB3.

The LED RAM data can be allocated as follows:

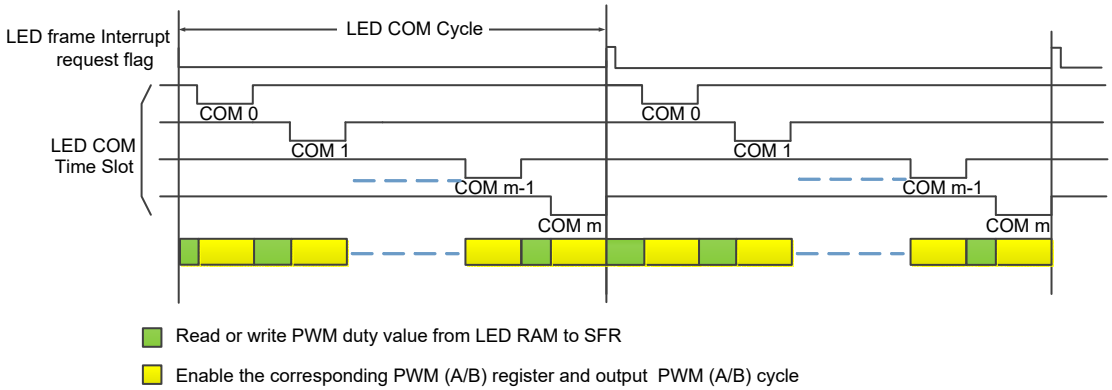


LED Memory Allocation (n=0~3, m=0~15)

LED Operation

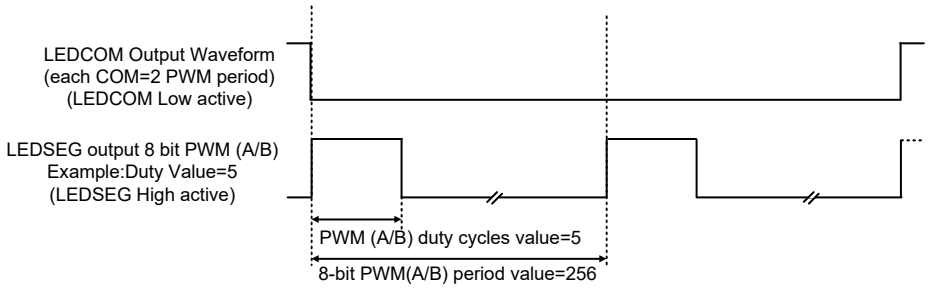
The LED function can be configured to operate in manual mode or auto mode, which is determined by the LDCOME bit. If the LDCOME bit is set high to select the auto mode, the LED function will automatically complete the related PWM duty and LED COM output by hardware. If the LDCOME bit is cleared to zero, users can also control the PWM output in manual mode by using the LED PWM modules, PWM register enable signals and duty data. The COMDIR and SEGDIR bits are used to setup the LEDCOM and LEDSEG output polarity, special attention should be paid to this point in application programs.

The timing of hardware LED PWM, i.e. 2×(n+1) LED Segment PWM and (m+1) COM, is shown as follow, where the LEDCOM is low active and LEDSEG is high active:



LED PWM Timing Diagram (n=0~3, m=0~15)

The PWM Module PWM output waveform is as follows:



Module PWM Output Waveform

Hardware Auto Key Scan Function

The LED PWM also supports a Hardware Auto Key scan function, which can be enabled by setting the KSCE bit. This function allows the data on the key scan output ports KEYRm, the ports which are pin-shared with LEDCOMm functions or with only I/O functions, to be input through the key scan input pins KEYC0~KEYC7 during the LED COM output or Key Scan output duration, the input data on the KEYC0~KEYC7 pins will then be captured and stored into the KEYDATAx registers at the instant the LED COM output changed to an inactive state. This function can be configured using the related register bits, the KSCE, PHRC, TDELAY3~TDELY0 and KEYSR4~KEYSR0 bits, as well as the KEYDATAx registers used for scan input data storage.

The related timings of LED COM output and Key Scan are shown in the following diagrams:

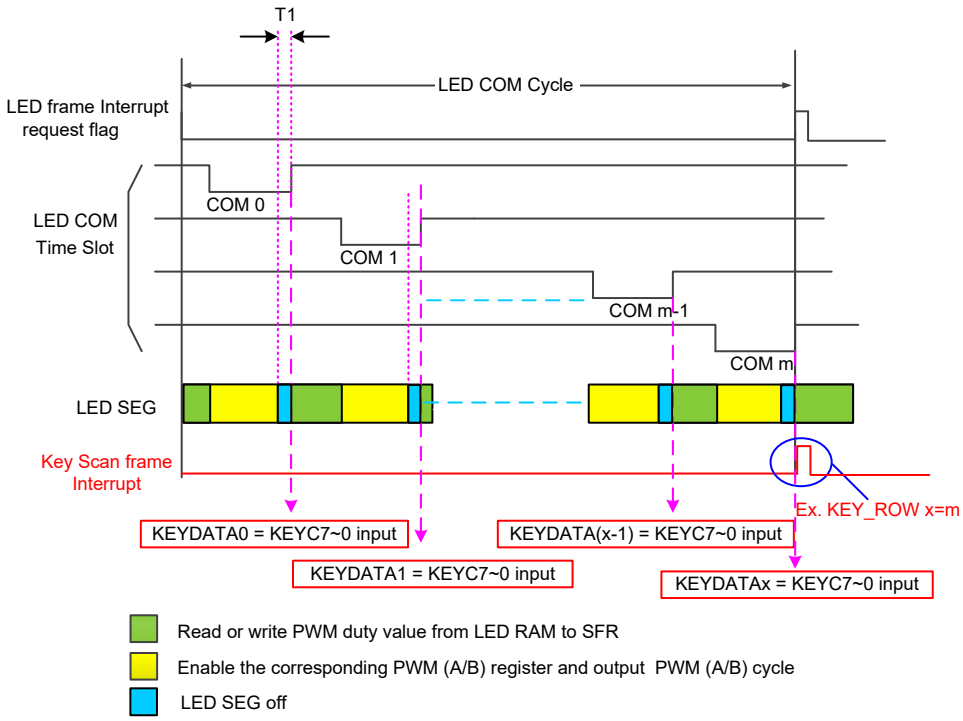
Condition (1):

KSCE=1; Enable hardware Key Scan function

PHRC=0; Disable hardware Key Scan input port pull high resistor

T1=TDELAY3~0

KEYSR4~0 (x)=LDCOM3~0 (m)



Condition (2):

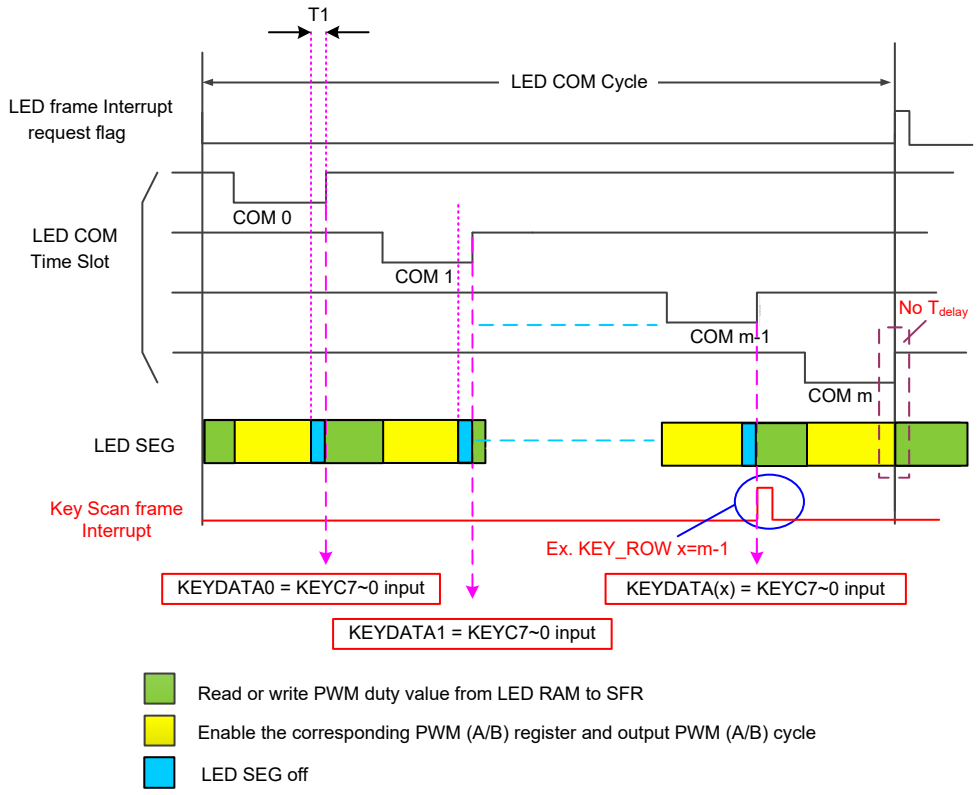
KSCE=1; Enable hardware Key Scan function

PHRC=0; Disable hardware Key Scan input port pull high resistor

KEYSR4~0 (x) < LDCOM3~0 (m)

T1=TDELAY3~0

Ex. x=m-1



Condition (3):

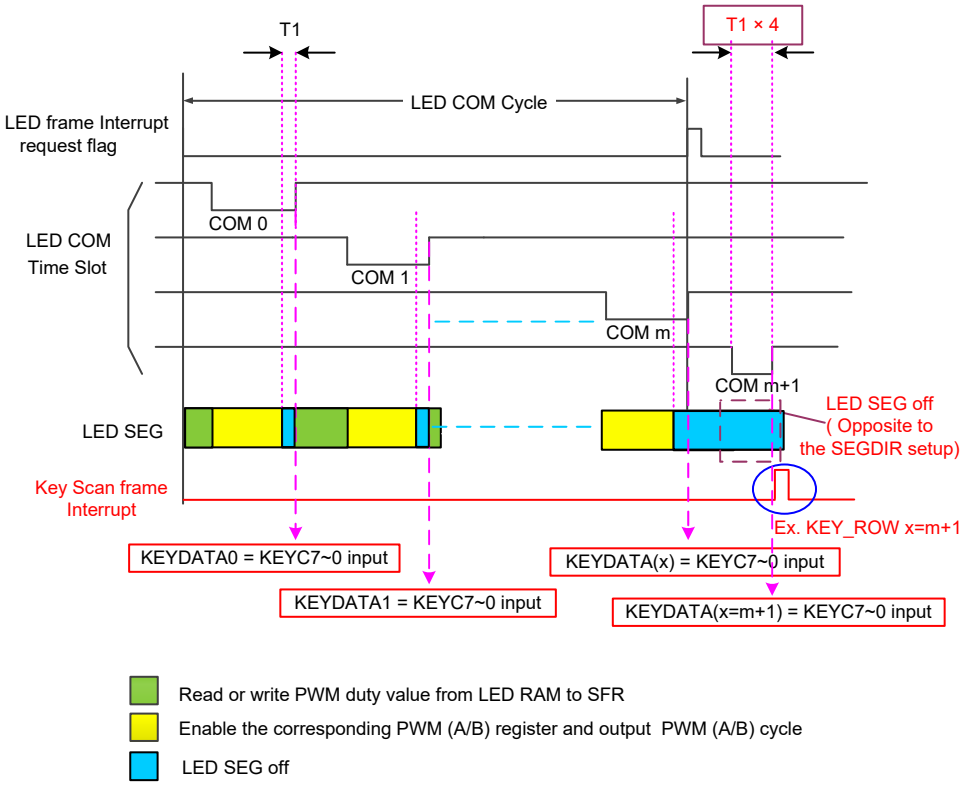
KSCE=1; Enable hardware Key Scan function

PHRC=0; Disable hardware Key scan input port pull high resistor

KEYSR4~0 (x) > LDCOM3~0 (m)

T1=TDELAY3~0

Ex. x=m+1



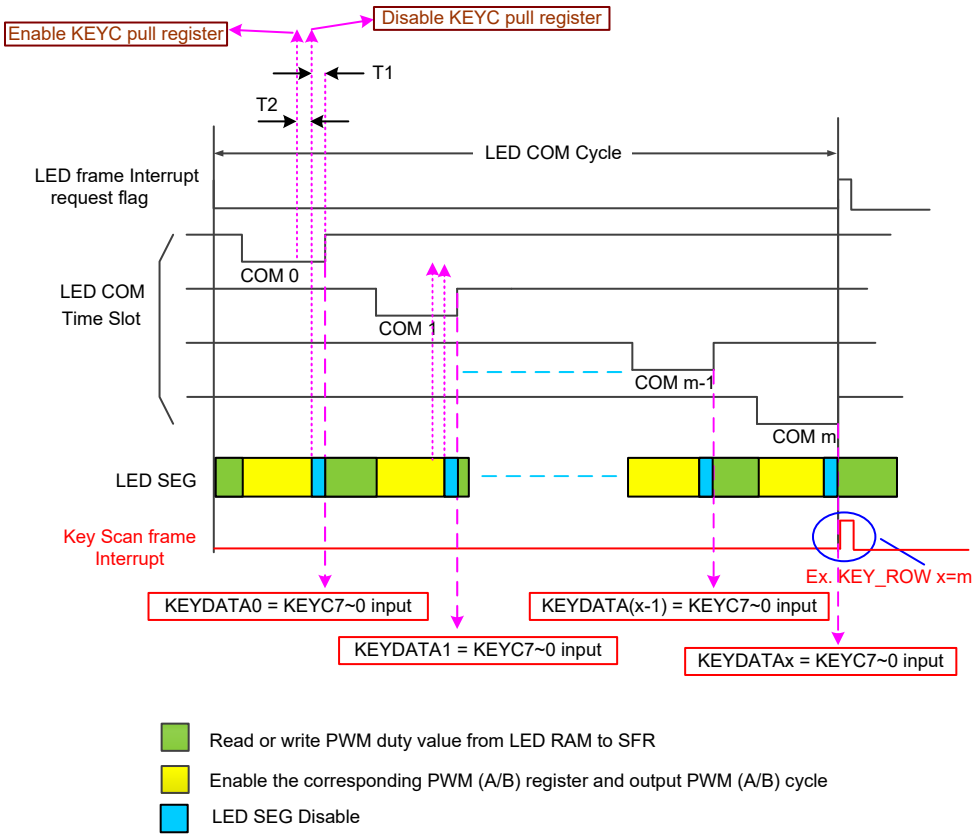
Condition (4):

KSCE=1; Enable hardware Key Scan function

PHRC=1; Enable hardware Key Scan input port pull high resistor

T1=TDELAY3~0; T2=TPHRE1~0

KEYSR4~0 (x)=LDCOM3~0 (m)



Condition (5):

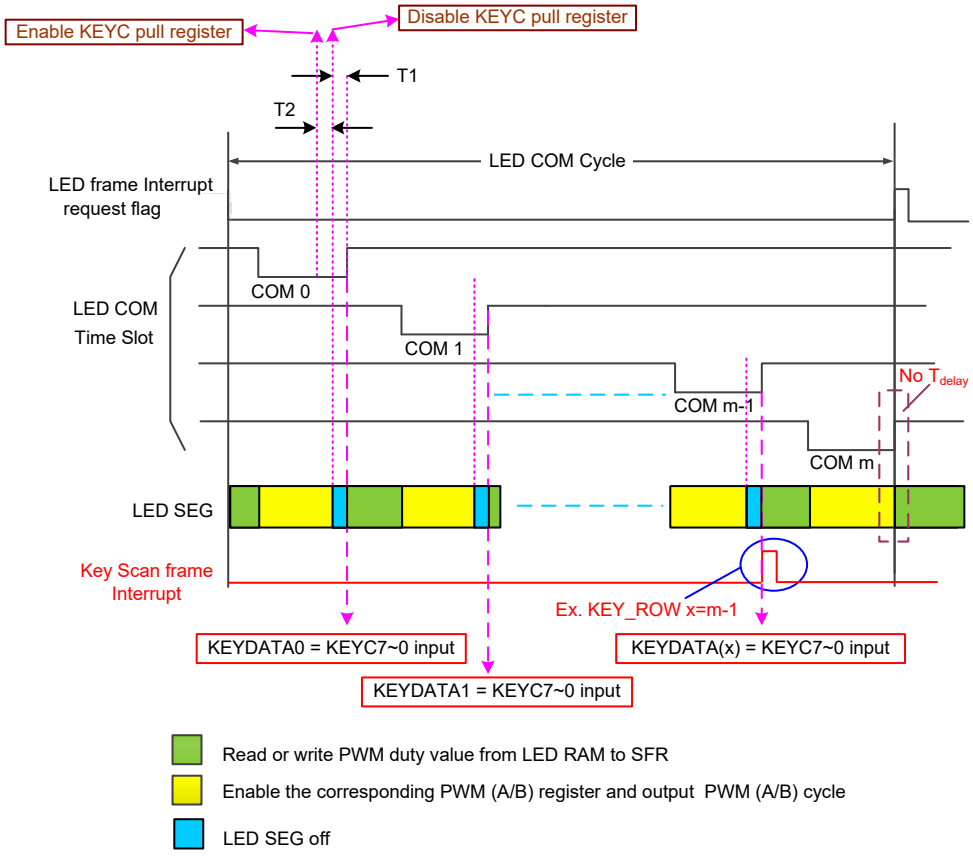
KSCE=1; Enable hardware Key Scan function

PHRC=1; Enable hardware Key Scan input port pull high resistor

KEYSR4~0 (x) < LDCOM3~0 (m)

T1=TDELAY3~0; T2=TPHRE1~0

Ex. x=m-1;



Condition (6):

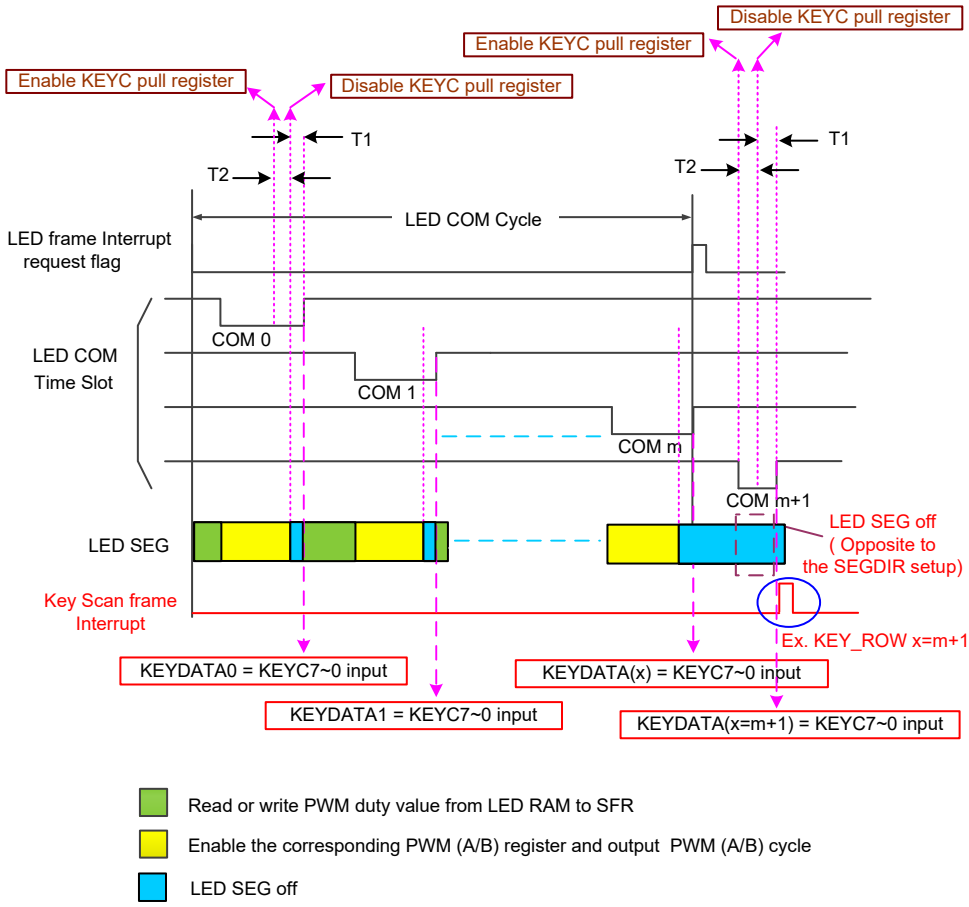
KSCE=1; Enable hardware Key Scan function;

PHRC=1; Enable hardware Key Scan input port pull high resistor control

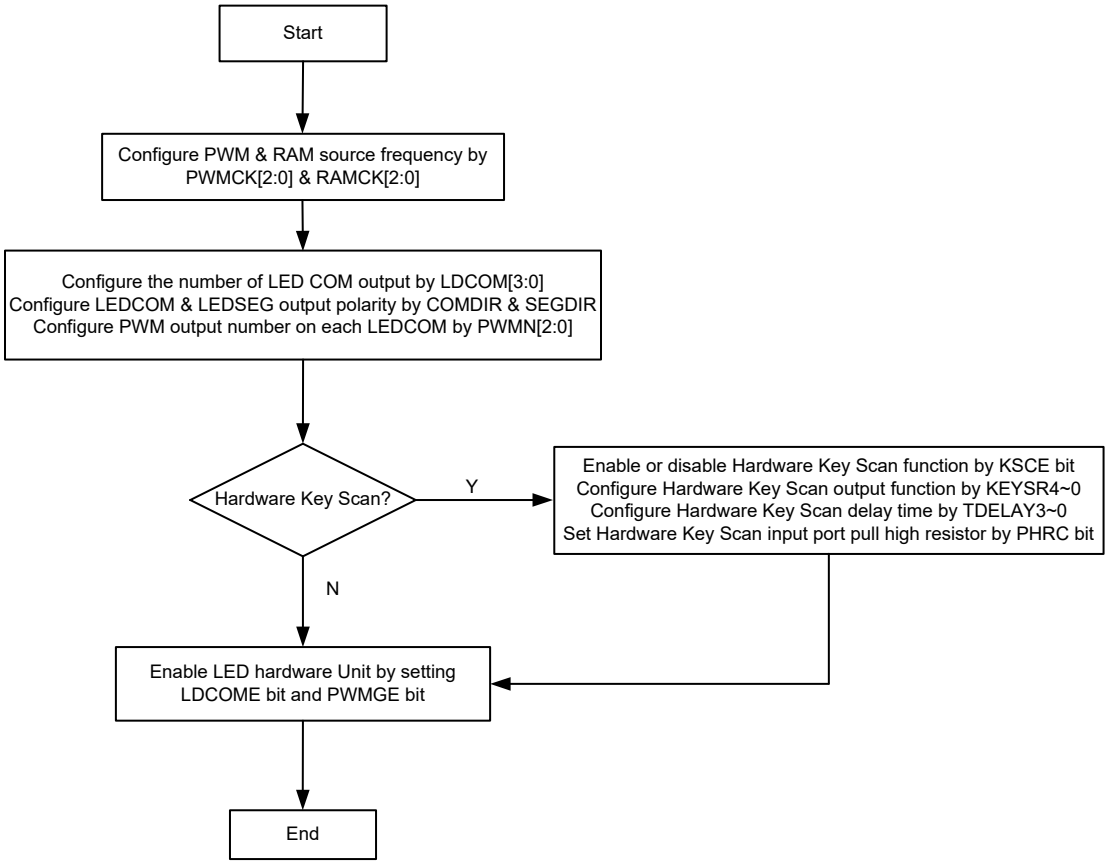
T1=TDELAY3~0; T2=TPHRE1~0

KEYSR4~0 (x) > LDCOM3~0 (m)

Ex. x=m+1



The global LED PWM function configuration flowchart is shown as below:



PWM Output Control Flow Chart

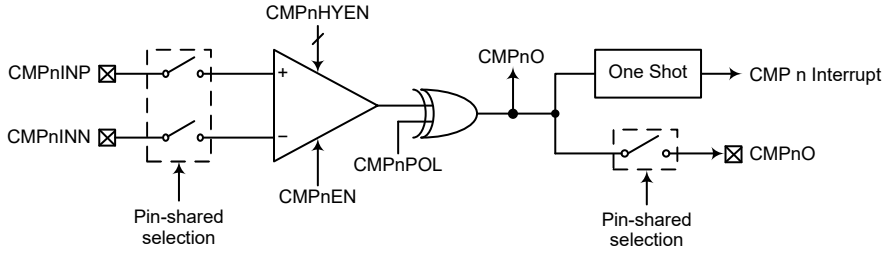
LED Interrupt

The LED function has two interrupt output signals, LED PWM frame interrupt and Key Scan frame interrupt. After one frame, i.e. 2(n+1)×(m+1) LED segment PWM signals, is completed by the hardware, an LED PWM frame interrupt signal will be generated to inform the MCU. It should be noted that when the LED PWM is disabled by application program, it will not be disabled until the hardware completes the current PWM frame and sends out an interrupt signal.

When the Hardware Key Scan function is enabled, a Key Scan frame interrupt signal will be generated to inform the MCU when the Key Scan frame is completed. Refer to the Interrupt section for more detailed information.

Comparators – HT68FB541 Only

Two independent analog comparators are contained within the HT68FB541 device. These functions offer flexibility via their register controlled features such as power-down, polarity select, hysteresis etc. In sharing their pins with normal I/O pins the comparators do not waste precious I/O pins if there functions are otherwise unused.



Comparators (n=0~1)

Comparator Operation

The HT68FB541 device contains two comparator functions which are used to compare two analog voltages and provide an output based on their difference.

Any pull-high resistors connected to the shared comparator input pins will be automatically disconnected when the comparator is enabled. As the comparator inputs approach their switching level, some spurious output signals may be generated on the comparator output due to the slow rising or falling nature of the input signals. This can be minimised by selecting the hysteresis function will apply a small amount of positive feedback to the comparator. Ideally the comparator should switch at the point where the positive and negative inputs signals are at the same voltage level. However, unavoidable input offsets introduce some uncertainties here. The hysteresis function, if enabled, also increases the switching offset value.

Comparator Registers

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

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

Bit	7	6	5	4	3	2	1	0
Name	—	CMPnEN	CMPnPOL	CMPnO	—	—	—	CMPnHYEN
R/W	—	R/W	R/W	R	—	—	—	R/W
POR	—	0	0	0	—	—	—	1

Bit 7 Unimplemented, read as "0"

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

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

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

Bit 3~1 Unimplemented, read as "0"

Bit 0 **CMPnHYEN**: Comparator n hysteresis enable control
0: Disable
1: Enable
This is the comparator n hysteresis enable control bit and if set high will apply a limited amount of hysteresis to the comparator. The positive feedback induced by hysteresis reduces the effect of spurious switching near the comparator threshold.

Comparator Interrupt

Each comparator also possesses its own interrupt function. When any one of the output bits changes state, its relevant interrupt flag will be set, and if the corresponding interrupt enable bit is set, then a jump to its relevant interrupt vector will be executed. Note that it is the changing state of the CMPnO bit and not the output pin which generates an interrupt. If the microcontroller is in the SLEEP or IDLE Mode and the Comparator is enabled, then if the external input lines cause the Comparator output bit to change state, the resulting generated interrupt flag will also generate a wake-up. If it is required to disable a wake-up from occurring, then the interrupt flag should be first set high before entering the SLEEP or IDLE Mode.

Programming Considerations

If the comparator is enabled, it will remain active when the microcontroller enters the SLEEP or IDLE Mode, however as it will consume a certain amount of power, users may wish to consider disabling it before the SLEEP or IDLE Mode is entered.

As comparator pins are shared with normal I/O pins the I/O registers for these pins will be read as zero (port control register is "1") or read as port data register value (port control register is "0") if the comparator function is enabled.

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 devices contain several external interrupts and internal interrupts functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the EEPROM, Timer/event counters, Comparators (HT68FB541 only), LED PWM functions, SPI, LVD, Time Bases and the USB interface.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers falls into two categories. The first is the INTC0~INTC3 registers which setup the primary interrupts and the second 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 or 1
USB	USBE	USBF	—
USB start of frame	SOFE	SOFF	—
LED PWM frame	LEDE	LEDF	—
Key Scan frame	KSCIE	KSCIF	—
Time Bases	TBnE	TBnF	n=0 or 1
LVD	LVE	LVF	—
Timer/Event Counters	TnE	TnF	n=0 or 1
EEPROM	DEE	DEF	—
SPI	SPIE	SPIF	—
Comparators (HT68FB541 only)	CPnE	CPnF	n=0 or 1

Interrupt Register Bit Naming Conventions

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	TB1F	TB0F	KSCIF	LEDF	TB1E	TB0E	KSCIE	LEDE
INTC2	DEF	T1F	T0F	LVF	DEE	T1E	T0E	LVE
INTC3 (HT68FB541)	CP1F	CP0F	SOFF	SPIF	CP1E	CP0E	SOFE	SPIE
INTC3 (HT68FB571)	—	—	SOFF	SPIF	—	—	SOFE	SPIE

Interrupt Register List

• **INTEG Register**

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

Bit 7~4 Unimplemented, read as "0"

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

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

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	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 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 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	TB1F	TB0F	KSCIF	LEDF	TB1E	TB0E	KSCIE	LEDE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TB1F**: Time base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TB0F**: Time base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **KSCIF**: Key scan frame interrupt request flag
0: No request
1: Interrupt request

Bit 4 **LEDF**: LED PWM frame interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TB1E**: Time base 1 interrupt control
0: Disable
1: Enable

Bit 2 **TB0E**: Time base 0 interrupt control
0: Disable
1: Enable

Bit 1 **KSCIE**: Key scan frame interrupt control
0: Disable
1: Enable

Bit 0 **LEDE**: LED PWM frame interrupt control
0: Disable
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	DEF	T1F	T0F	LVF	DEE	T1E	T0E	LVE
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 **DEF**: Data EEPROM interrupt request flag
0: No request
1: Interrupt request

Bit 6 **T1F**: Timer/event counter 1 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **T0F**: Timer/event counter 0 interrupt request flag
0: No request
1: Interrupt request

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

Bit 3 **DEE**: Data EEPROM interrupt control
0: Disable
1: Enable

Bit 2 **T1E**: Timer/event counter 1 interrupt control
0: Disable
1: Enable

Bit 1 **T0E**: Timer/event counter 0 interrupt control
0: Disable
1: Enable

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

• INTC3 Register – HT68FB541

Bit	7	6	5	4	3	2	1	0
Name	CP1F	CP0F	SOFF	SPIF	CP1E	CP0E	SOFE	SPIE
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 **CP1F**: Comparator 1 interrupt request
0: No request
1: Interrupt request

Bit 6 **CP0F**: Comparator 0 interrupt request
0: No request
1: Interrupt request

Bit 5 **SOFF**: USB start-of-frame interrupt request flag
0: No request
1: Interrupt request

Bit 4 **SPIF**: SPI interrupt request flag
0: No request
1: Interrupt request

Bit 3 **CP1E**: Comparator 1 interrupt control
0: Disable
1: Enable

Bit 2 **CP0E**: Comparator 0 interrupt control
0: Disable
1: Enable

Bit 1 **SOFE**: USB start-of-frame interrupt control
0: Disable
1: Enable

Bit 0 **SPIE**: SPI interrupt control
0: Disable
1: Enable

• INTC3 Register – HT68FB571

Bit	7	6	5	4	3	2	1	0
Name	—	—	SOFF	SPIF	—	—	SOFE	SPIE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5 **SOFF**: USB start-of-frame interrupt request flag
0: No request
1: Interrupt request

Bit 4 **SPIF**: SPI interrupt request flag
0: No request
1: Interrupt request

Bit 3~2	Unimplemented, read as "0"
Bit 1	**SOFE**: USB start-of-frame interrupt control 0: Disable 1: Enable
Bit 0	**SPIE**: SPI interrupt control 0: Disable 1: Enable

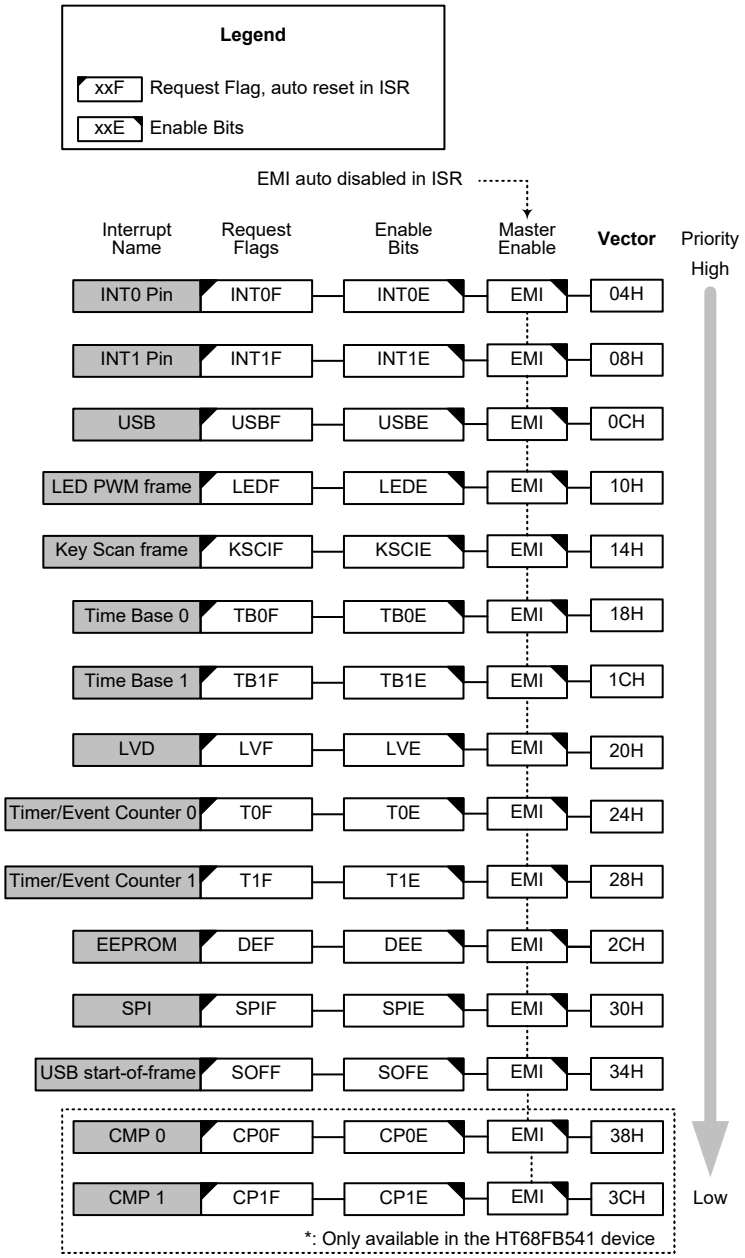
Interrupt Operation

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

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

The external interrupts are controlled by signal transitions on the pins INTn. 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 pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INTnE, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable

bit in the corresponding interrupt register has been set and the external interrupts pin are selected by the corresponding pin-shared function selection bits. The pins must also be setup as input by setting the corresponding bits 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 pins, 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 pins will remain valid even if the pin is used as an external interrupt input.

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

Timer/Event Counter Interrupts

A timer/event counter interrupt request will take place when the corresponding interrupt request flag, TnF, is set, which occurs when the timer/event counter overflows. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI and the corresponding timer interrupt enable bit TnE must first be set. When the interrupt is enabled, the stack is not full and a timer/event counter overflow occurs, a subroutine call to the relevant timer interrupt vector will take place. When the interrupt is serviced, the timer interrupt request flag TnF will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

USB Interrupt

Several USB conditions can generate a USB interrupt. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are the USB suspended, USB resumed, USB reset and USB endpoint FIFO access events. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and USB interrupt enable bit, USBE, must first be set. When the interrupt is enabled, the stack is not full and any of these conditions are created, a subroutine call to the USB interrupt vector, will take place. When the interrupt is serviced, the USB interrupt request flag, USBF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

USB Start of Frame Interrupt

A USB SOF interrupt request will take place when the correspongding interrupt request flag, SOFF, is set, which occurs when a USB start of frame signal is detected. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and USB start of frame interrupt enable bit, SOFE, must first be set. When the interrupt is enabled, the stack is not full and a USB start of frame signal is detected, a subroutine call to the respective interrupt vector will take place. When the USB SOF interrupt is serviced, the SOFF flag will be automatically cleared, the EMI bit will also be automatically cleared to disable other interrupts.

SPI Interrupt

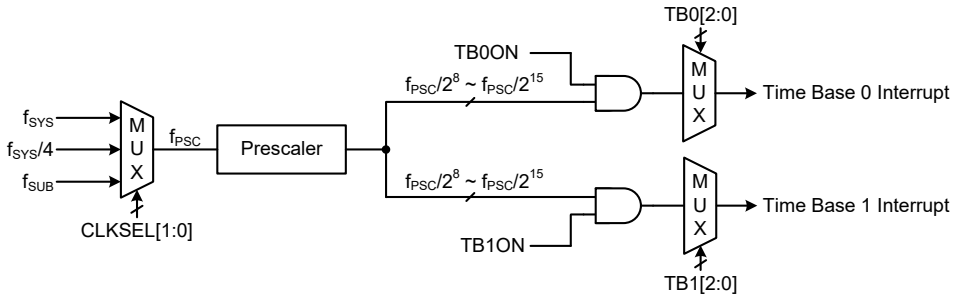
The serial peripheral Interface interrupt, also known as the SPI interrupt, will take place when the SPI interrupt request flag, SPIF, is set, which occurs when a byte of data has been received or transmitted by the SPI interface or an SPI incomplete transfer occurs. To allow the program to branch to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

interrupt flag, SPIF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Time Base Interrupts

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

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



Time Base Interrupts

• PSCR Register

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

Bit 7~2 Unimplemented, read as "0"

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

• TBnC Register (n=0~1)

Bit	7	6	5	4	3	2	1	0
Name	TBnON	—	—	—	—	TBn2	TBn1	TBn0
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TBnON**: Time Base n control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TBn2~TBn0**: Select Time Base n time-out period
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

LED PWM Frame Interrupt

An LED PWM frame interrupt request will take place when the LED PWM frame interrupt request flag, LEDF, is set, which occurs when one frame LED PWM is completed. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the LED PWM frame interrupt enable bit, LEDE, must first be set. When the interrupt is enabled, the stack is not full and one frame LED PWM is completed, a subroutine call to the LED PWM frame interrupt vector will take place. When the LED PWM frame interrupt is serviced, the LEDF flag will be automativally cleared, the EMI bit will also be automatically cleared to disable other interrupts.

Key Scan Frame Interrupt

A key scan frame interrupt request will take place when the key scan frame interrupt request flag, KSCIF, is set, which occurs when one key scan frame is completed. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the key scan frame interrupt enable bit, KSCIE, must first be set. When the interrupt is enabled, the stack is not full and one key scan frame is completed, a subroutine call to the key scan frame interrupt vector will take place. When the key scan frame interrupt is serviced, the KSCIF flag will be automativally cleared, the EMI bit will also be automatically cleared to disable other interrupts.

EEPROM Interrupt

An EEPROM interrupt request will take place when the EEPROM interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM interrupt enable bit, DEE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the respective EEPROM interrupt vector will take place. When the EEPROM interrupt is serviced, the DEF flag will be automatically cleared, the EMI bit will also be automatically cleared to disable other interrupts.

Comparator Interrupts – HT68FB541 Only

The comparator interrupts are controlled by the two internal comparators. A Comparator interrupt request will take place when the comparator interrupt request flag, CPnF, is set, a situation that will occur when the comparator output bit changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and comparator interrupt enable bit, CPnE, must first be set. When the interrupt is enabled, the stack is not full and the comparator inputs generate a comparator output bit transition, a subroutine call to the comparator interrupt vector, will take place. When the interrupt is serviced, the comparator interrupt request flag, CPnF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

LVD Interrupt

An LVD interrupt request will take place when the LVD interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Low Voltage Detection 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 Detection interrupt is serviced, the LVF flag will be automatically cleared, the EMI bit will also be automatically cleared to disable other interrupts.

Interrupt Wake-up Function

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

Programming Considerations

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

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

Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

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

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

Low Voltage Detector – LVD

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

LVD Register

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

- **LVDC Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **LVDO**: LVD output flag
0: No Low Voltage Detected
1: Low Voltage Detected

Bit 4 **LVDEN**: Low voltage detector enable control
0: Disable
1: Enable

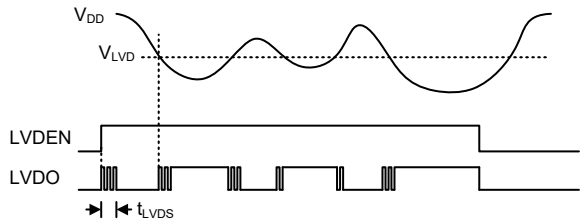
Bit 3 **VBGEN**: Bandgap voltage output enable control
0: Disable
1: Enable
Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VBGEN bit is set to 1.

Bit 2~0 **VLVD2~VLVD0**: LVD voltage selection
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.0V and 4.0V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. When the device is in the SLEEP mode,

the low voltage detector will be disabled even if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} falls below the preset LVD voltage. This will cause the device to wake-up from the IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before the device enters the IDLE Mode.

Application Description

The HT68FB541 device is mainly used for full speed USB gaming mouse with flow lighting effects. In addition to supporting a 1ms report rate, the device can be applied to drive up to 8 RGB LEDs. Due to the increased drive I/O current, the external RGB LEDs can be arranged as pairwise parallel connection to produce excellent display effects, which can be applied for the mouse side flow lighting effect implementation. The HT68FB571 device is mainly used for full speed USB gaming keyboard with monochrome and single light features. The device supports a 1ms report rate and the ability of driving up to 128PSC monochrome LEDs or 32 RGB LEDs.

Both of the devices provide the required hardware functions, such as LED PWM Scan and Key Scan functions, which are introduced as below.

Functional Description

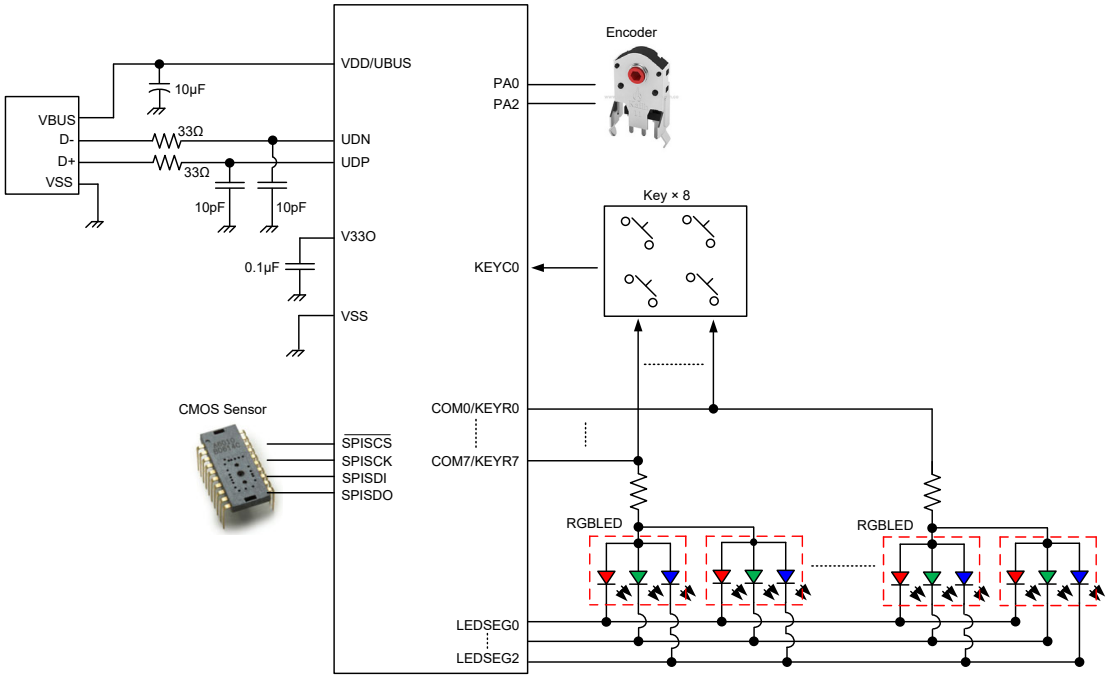
a. The devices contain a Full Speed USB function, which has integrated 3.3V LDO for the provision of the required 3.3V voltage for USB signals, the required 1.5kΩ resistor for USB Full Speed function is also provided.

b. The HT68FB541 device uses an 8×3 matrix scanning method while the HT68FB571 device uses a 16×8 matrix scanning method, the integrated hardware 8-bit LED PWM Driver will complete scan function by hardware, which can achieve a high frame rate, only several current-limiting resistors are required for external circuit.

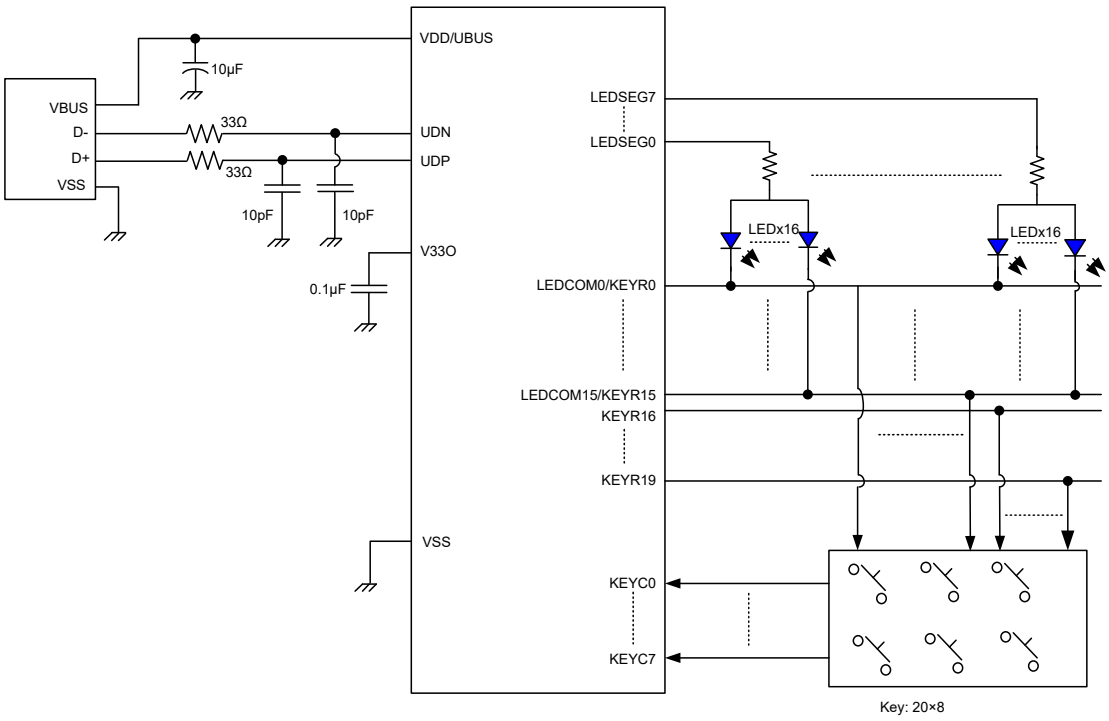
c. The keyboard required key scan function is also implemented by hardware. For the HT68FB541 device, the key scan size is 8×1, in which the key scan output 0~7 are all pin-shared with LED COM outputs. For the HT68FB571 device the key scan size is 20×8, in which the key scan output 0~15 are pin-shared with LED COM outputs, while the key scan output 16~19 are pin-shared with only I/O functions. The key scan result can be obtained by reading the KEYDATAx register contents.

Hardware Circuit

HT68FB541 Application Circuit for Mouse



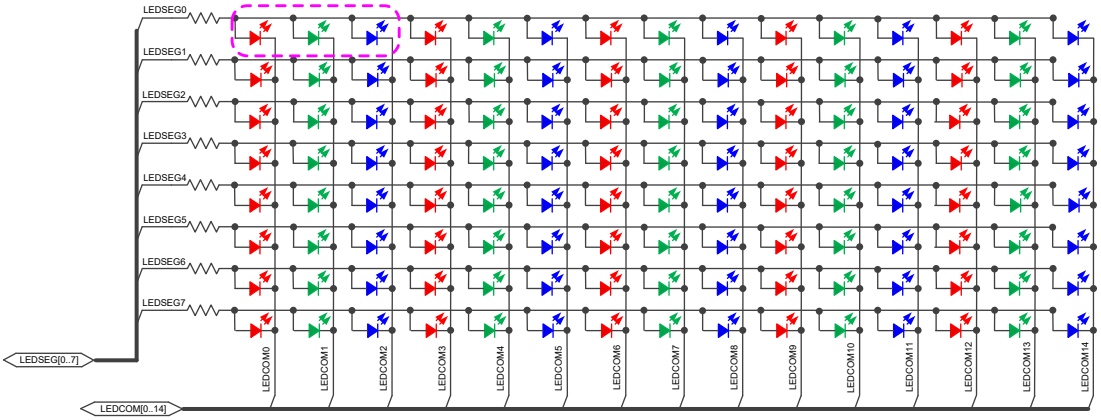
HT68FB571 Application Circuits for LED Keyboard



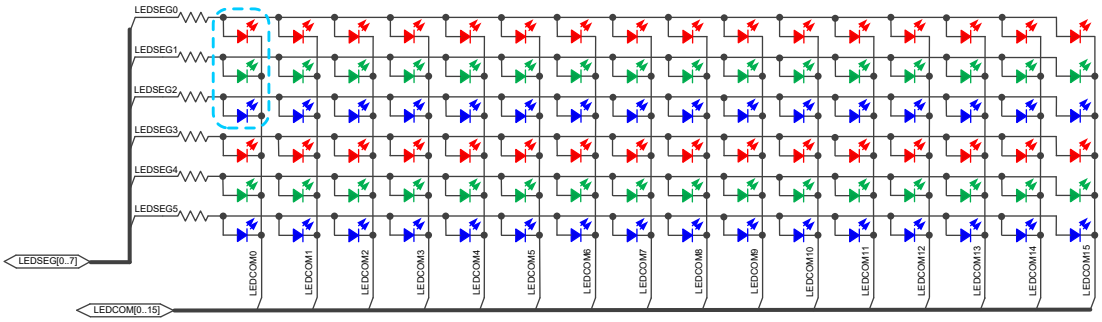
Hardware Circuits for RGB LED Applications

The HT68FB571 can also be applied to RGB LED applications. The following diagrams show the recommended matrix circuits when the device is used in the common anode and common cathode RGB LEDs, in which the LEDSEG output polarity is active high and the LEDCOM output polarity is active low.

a. The following diagram shows the recommended LED matrix circuit of common anode RGB LED. The LEDs framed by a dash line means a common anode RGB LED. The HT68FB571 can be applied to up to 40 common anode RGB LEDs.



b. The following diagram shows the recommended LED matrix circuit of common cathode RGB LED. The LEDs framed by a dash line means a common cathode RGB LED. The HT68FB571 can be applied to up to 32 common cathnode RGB LEDs.



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

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

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

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

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

Extended Instruction Set

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

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

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

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

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

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

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

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

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

Affected flag(s) C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s): C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: $ACC \leftarrow ACC - [m] - \overline{C}$

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

SBC A, x Subtract immediate data from ACC with Carry

Description: The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: $ACC \leftarrow ACC - [m] - \overline{C}$

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

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: $[m] \leftarrow ACC - [m] - \overline{C}$

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

SDZ [m] Skip if decrement Data Memory is 0

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

Operation:
[m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

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

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

Operation:
ACC ← [m] − 1
Skip if ACC=0

Affected flag(s): None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m]≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

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

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

Description The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

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

Operation Skip if [m].i=0

Affected flag(s) None

TABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Extended Instruction Definition

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

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

LADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

LADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″AND″ [m]

Affected flag(s) Z

LCLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

LCLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

LCPL [m] Complement Data Memory

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

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

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

LDEC [m] Decrement Data Memory

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

Operation $[m] \leftarrow [m] - 1$

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

LINC [m] Increment Data Memory

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

Operation $[m] \leftarrow [m] + 1$

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] + 1$

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

LMOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

LRL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

LRLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

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

Affected flag(s) C

LRLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) C

LRR [m] Rotate Data Memory right

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

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

Affected flag(s) None

LRRA [m] Rotate Data Memory right with result in ACC

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

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

Affected flag(s) None

LRRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

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

Affected flag(s) C

LRRCA [m] Rotate Data Memory right through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) C

LSBC A,[m] Subtract Data Memory from ACC with Carry

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m] − $\overline{C}$

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

LSBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m] − $\overline{C}$

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

LSDZ [m] Skip if decrement Data Memory is 0

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

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

LSET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

LSET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

LSIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the content of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m] ≠ 0

Affected flag(s) None

LSUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

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

LSUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

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

LSWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

LSWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

LSZ [m] Skip if Data Memory is 0

Description The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

LSZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

LSZ [m].i Skip if bit i of Data Memory is 0

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

Operation Skip if [m].i=0

Affected flag(s) None

LTABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LXOR A,[m] Logical XOR Data Memory to ACC

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

LXORM A,[m] Logical XOR ACC to Data Memory

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

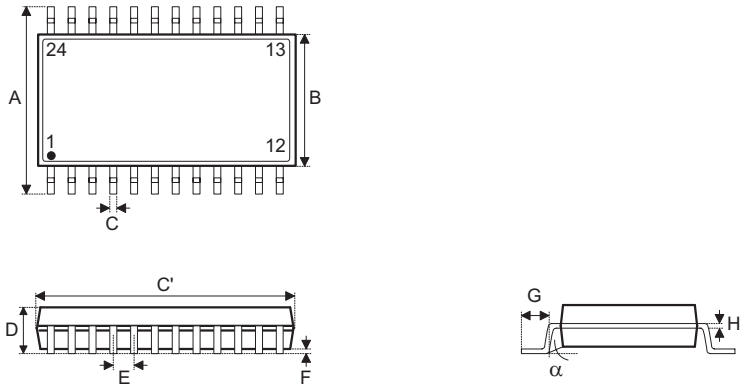
Package Information

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

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

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

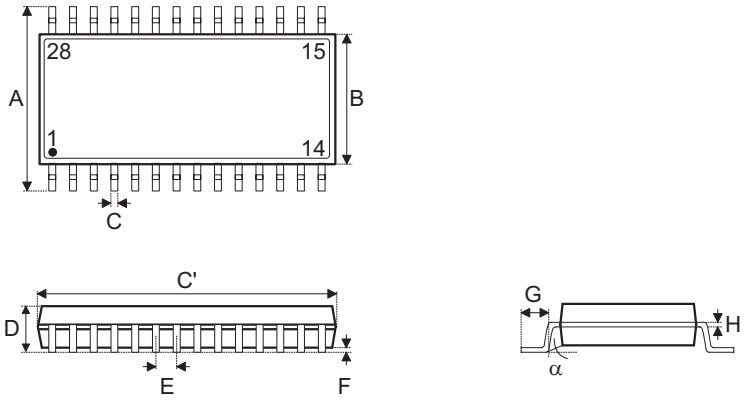
24-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.341 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.00 BSC	—
B	—	3.90 BSC	—
C	0.20	—	0.30
C’	—	8.66 BSC	—
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

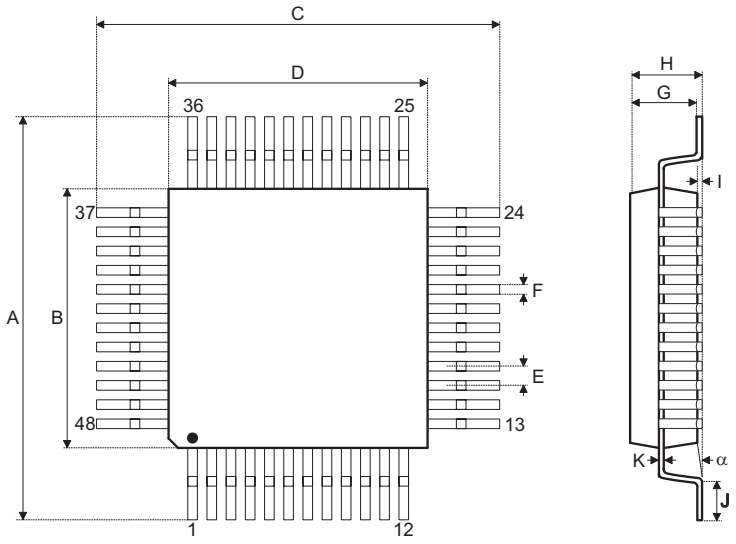
28-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C'	—	0.390 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.00 BSC	—
B	—	3.90 BSC	—
C	0.20	—	0.30
C'	—	9.90 BSC	—
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

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°