Voice Peripheral MCU

HT68FV022/HT68FV024

Revision: V1.50 Date: July 27, 2023

www.holtek.com

Features

CPU Features

- Operating Voltage
  - f_{SYS}=8MHz: 2.3V~5.5V
  - f_{SYS}=12MHz: 2.3V~5.5V
  - f_{SYS}=16MHz: 3.0V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 8/12/16MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 1K×14
- RAM Data Memory: 64×8
- Voice Flash Memory: HT68FV022: 16Mbit; HT68FV024: 32Mbit
- Watchdog Timer function
- 5 bidirectional I/O lines
- One external interrupt line shared with I/O pin
- One 8-bit programmable Timer/Event Counter with overflow interrupt and prescaler
- Single Time Base function for generation of fixed time interrupt signal
- Audio PWM driver
- Integrated 3.0V LDO
- Low voltage reset function – LVR
- Support Peripheral IC mode
- Package types: 8-pin SOP, 16-pin NSOP

Development Tools

For rapid product development and to simplify device parameter setting, Holtek has provided relevant development tools which users can download from the following link:

https://www.holtek.com/voice_mcu_workshop

General Description

The devices are Flash Memory 8-bit high performance RISC architecture microcontrollers, which are designed for various voice application terminal products such as smart home appliances, consumer electronic products, etc.

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 up to 32Mbit Voice Flash Memory.

A single extremely flexible Timer/Event Counter provides timing, event counting and pulse wide measurement functions. In addition, an internal LDO function provides power to the Voice Flash Memory. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of internal high and low speed oscillators are provided, the two fully integrated system oscillators 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 devices integrate a audio PWM driver, for which the devices have a digital programmable volume control with a wide range. The devices also support the Peripheral IC mode, implementing voice editing together with Holtek Voice MCU Workshop. The inclusion of flexible I/O programming features, Time Base function, along with many other features ensure that the devices will find excellent use for voice application terminal products.

Selection Table

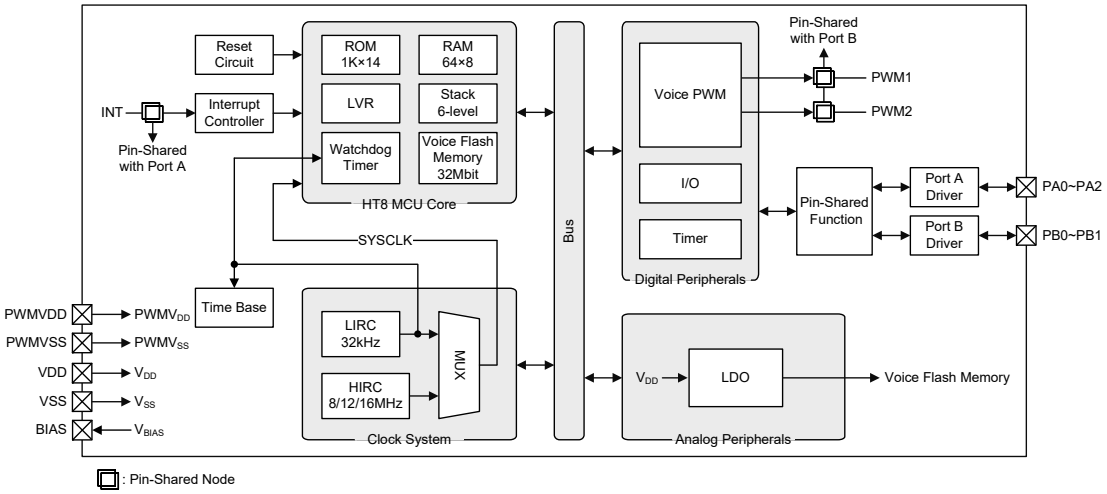
Most features are common to all devices and the main features distinguishing them are Voice Flash Memory capacity and maximum voice time. The following table summarises the main features of each device.

Part No.	V_{DD}	Program Memory	Data Memory	Voice Flash Memory	Maximum Voice Time
HT68FV022	2.3V~5.5V	1K×14	64×8	16Mbit	400s
HT68FV024				32Mbit	800s

Part No.	I/O	External Interrupt	Time Base	Timer	Stack	Package
HT68FV022	5	1	1	8-bit×1	6	8SOP 16NSOP
HT68FV024						

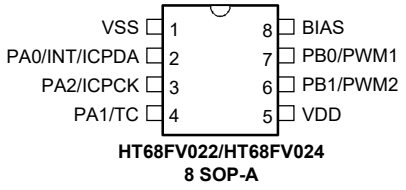
Note: As devices exist in more than one package format, the table reflects the situation for the package with the most pins.

Block Diagram

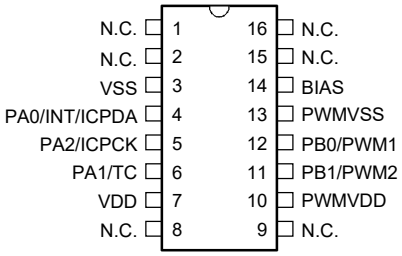


Note: The figure illustrates the Block Diagram of the device with maximum features, the functional differences between the series of devices are provided in the Selection Table.

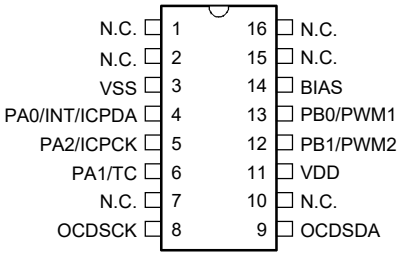
Pin Assignment



HT68FV022/HT68FV024
8 SOP-A



HT68FV022/HT68FV024
16 NSOP-A



HT68VV022/HT68VV024
16 NSOP-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 as OCDS dedicated pins and as such only available for the HT68VV022/V024 device which is the OCDS EV chip for the HT68FV022/V024 device.

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

Pin 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 tables will be available on smaller package sizes.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT/ICPDA	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT	INTEG INTC0	ST	—	External interrupt input
	ICPDA	—	ST	CMOS	ICP data/address
PA1/TC	PA1	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	TC	—	ST	—	Timer/Event Counter clock input
PA2/ICPCK	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	ICPCK	—	ST	—	ICP clock pin
PB0/PWM1	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PWM1	PBS0	—	CMOS	PWM driver output 1
PB1/PWM2	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PWM2	PBS0	—	CMOS	PWM driver output 2
BIAS	BIAS	—	—	PWR	Supply IC basic power
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground
PWMVDD	PWMVDD	—	PWR	—	PWM positive power supply
PWMVSS	PWMVSS	—	PWR	—	PWM negative power supply, ground
OCDSDA	OCDSDA	—	ST	CMOS	OCDS data/address, for EV chip only
OCDSCK	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
N.C.	N.C.	—	—	—	No connected

Legend: I/T: Input type;
OPT: Optional by register option;
ST: Schmitt Trigger input;
O/T: Output type;
PWR: Power;
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..... -60°C to 150°C

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

I_{OL} Total 150mA

I_{OH} Total ..... -200mA

Total Power Dissipation 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the devices. Functional operation of the 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}	Operating Voltage – HIRC	f_{SYS}=8MHz	2.3	—	5.5	V
		f_{SYS}=12MHz	2.3	—	5.5	V
		f_{SYS}=16MHz	3.0	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32kHz	2.3	—	5.5	V

Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2.3V	f_{SYS}=32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	2.3V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	
		2.3V	f_{SYS}=12MHz	—	1.0	1.4	mA
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	
		3V	f_{SYS}=16MHz	—	1.5	3.0	mA
		5V		—	2.5	5.0	

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

1. Any digital inputs are 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=-40°C~85°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.3V	WDT off	—	0.11	0.15	2.00	μA
		3V		—	0.11	0.15	2.00	
		5V		—	0.18	0.38	2.90	
	IDLE0 Mode – LIRC	2.3V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE1 Mode – HIRC	2.3V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	
		2.3V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	540	750	900	
		5V		—	800	1200	1440	
		3V	f_{SUB} on, f_{SYS}=16MHz	—	0.80	1.20	1.44	mA
		5V		—	1.4	2.0	2.4	

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

1. Any digital inputs are 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 either 3V or 5V.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.3V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	
	12MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	12	+1%	MHz
			-40°C~85°C	-2%	12	+2%	
		2.3V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C~85°C	-3%	12	+3%	
	16MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	16	+1%	MHz
			-40°C~85°C	-2%	16	+2%	
		3.0V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C~85°C	-3%	16	+3%	

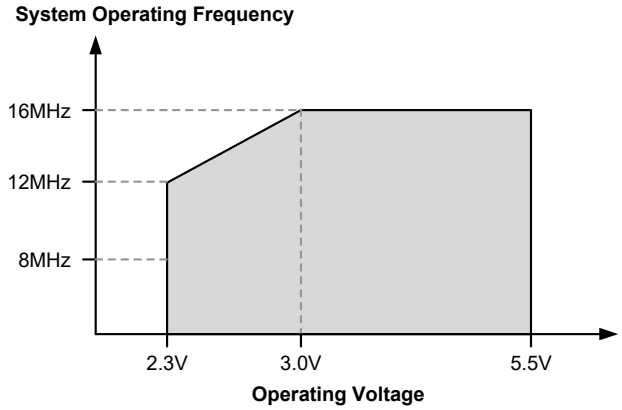
Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.3V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.
3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

Low Speed Internal Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency (Writer Trim)	3V/5V	25°C	-1.0%	32	+1.0%	kHz
	LIRC Frequency	2.3V~3.6V (trim @3V)	-10°C~50°C	-4.0%	32	+4.0%	
			-40°C~85°C	-6.0%	32	+6.0%	
		3.3V~5.5V (trim @5V)	-10°C~50°C	-4.0%	32	+4.0%	
			-40°C~85°C	-6.0%	32	+6.0%	
		2.3V~5.5V (trim @3V)	-40°C~85°C	-6.0%	32	+6.0%	
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 Time (Wake-up from Condition where f_{SYS} is off)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time (Wake-up from Condition where f_{SYS} is on)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	System Speed Switch Time (FAST to SLOW Mode or SLOW to FAST Mode)	—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
t_{RSTD}	System Reset Delay Time (Reset source from Power-on reset or LVR Hardware Reset)	—	RR_{POR}=5V/ms	14	16	18	ms
	System Reset Delay Time (LVRC/WDTC Register Software Reset)	—	—				
	System Reset Delay Time (WDT Overflow Reset)	—	—	14	16	18	
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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	VDD Power Supply	—	—	2.3	—	5.5	V
V_{IL}	Input Low Voltage for I/O Ports except PB0~PB1 Pins	5V	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports except PB0~PB1 Pins	5V	—	3.5	—	5.0	V
		–		$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Ports except PB0~PB1 Pins	3V	$V_{OL}=0.1V_{DD}$	5	10	—	mA
		5V		10	20	—	
I_{OH}	Source Current for I/O Ports except PB0~PB1 Pins	3V	$V_{OH}=0.9V_{DD}$	-2.5	-5.0	—	mA
		5V		-5	-10	—	
R_{PH}	Pull-high Resistance for I/O Ports except PB0~PB1 Pins (Note)	3V	—	20	60	100	kΩ
		5V		10	30	50	
I_{LEAK}	Input Leakage Current for I/O Ports	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{TC}	TC Input Pin Minimum Pulse Width	—	—	25	—	—	ns
t_{INT}	Interrupt Pin Minimum Pulse Width	—	—	0.3	—	—	μs

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

Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Flash Program Memory							
t_{DEW}	Erase/Write Cycle Time	—	—	—	2	3	ms
I_{DDPGM}	Programming/Erase Current on V_{DD}	—	—	—	—	5.0	mA
E_P	Cell Endurance	—	—	10K	—	—	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 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 1.7V	-5%	1.7	+5%	V
			LVR enable, voltage select 2.2V	-5%	2.2	+5%	
			LVR enable, voltage select 2.55V	-5%	2.55	+5%	
			LVR enable, voltage select 3.15V	-5%	3.15	+5%	
			LVR enable, voltage select 3.8V	-5%	3.8	+5%	
t_{LVR}	Minimum Low Voltage Width to Reset	—	TLVR[1:0]=00B	120	240	480	μs
			TLVR[1:0]=01B	0.5	1.0	2.0	ms
			TLVR[1:0]=10B	1	2	4	
			TLVR[1:0]=11B	2	4	8	
I_{LVR}	Additional Current for LVR Enable	3V	LVR enable, V_{LVR}=2.2V	—	—	10	μA
		5V		—	8	15	

Voice PWM Driver Characteristics

Audio PWM Driver Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB}	Standby Current	5V	PWM_EN=0; PLL_EN=0; DRV_EN=0	—	—	2	μA
t_{PLLS}	PLL Stable Time	—	PLL_EN=0 to PLL_EN=1	—	40	60	μs
I_{PLL}	Operating Current	—	PLL Enable (PLL_EN=1)	—	2	3	mA
I_{OL}	PWM1/PWM2 Sink Current	3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0000B	4.2	6.0	—	mA
		5V		10.5	15.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0001B	9.1	13.0	—	mA
		5V		20.3	29.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0010B	13.3	19.0	—	mA
		5V		29.4	42.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0011B	17.5	25.0	—	mA
		5V		38.5	55.0	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OL}	PWM1/PWM2 Sink Current	3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0100B	20.3	29.0	—	mA
		5V		44.1	63.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0101B	26.6	38.0	—	mA
		5V		56.7	81.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0110B	30.1	43.0	—	mA
		5V		64.4	92.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0111B	33.1	48.0	—	mA
		5V		71.4	102.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1000B	40.6	58.0	—	mA
		5V		84	120	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1001B	46.9	67.0	—	mA
		5V		95.9	137.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1010B	53.2	76.0	—	mA
		5V		107.8	154.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1011B	58.1	83.0	—	mA
		5V		117.6	168.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1100B	72	90	—	mA
		5V		144	180	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1101B	87.2	109.0	—	mA
		5V		170.4	213.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1110B	116.8	146.0	—	mA
		5V		217.6	272.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1111B	144	180	—	mA
		5V		256.8	321.0	—	mA
I_{OH}	PWM1/PWM2 Source Current	3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0000B	-4.2	-6.0	—	mA
		5V		-11.2	-16.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0001B	-9.1	-13.0	—	mA
		5V		-21.7	-31.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0010B	-13.3	-19.0	—	mA
		5V		-31.5	-45.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0011B	-17.5	-25.0	—	mA
		5V		-40.6	-58.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0100B	-19.6	-28.0	—	mA
		5V		-40.6	-66.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0101B	-19.6	-37.0	—	mA
		5V		-59.5	-85.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0110B	-29.4	-42.0	—	mA
		5V		-67.2	-96.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0111B	-32.9	-47.0	—	mA
		5V		-73.5	-105.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1000B	-39.2	-56.0	—	mA
		5V		-86.1	-123.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1001B	-44.8	-64.0	—	mA
		5V		-97.3	-139.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1010B	-50.4	-72.0	—	mA
		5V		-109.2	-156.0	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH}	PWM1/PWM2 Source Current	3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1011B	-55.3	-79.0	—	mA
		5V		-117.6	-168.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1100B	-68	-85	—	mA
		5V		-143.2	-179.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1101B	-81.6	-102.0	—	mA
		5V		-167.2	-209.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1110B	-106.4	-133.0	—	mA
		5V		-208	-260	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1111B	-128	-160	—	mA
		5V		-240	-300	—	mA

PWM/PB0/PB1 Driver Characteristics

Ta=-40°C~85°C

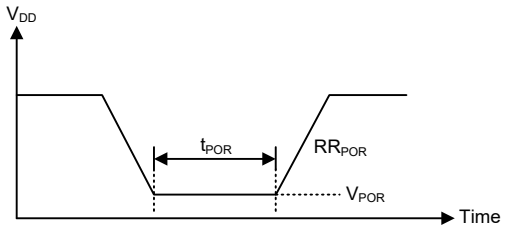
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for PB0~PB1 Pins	5V	—	0	—	2.0	V
		—		0	—	0.4V_{DD}	
V_{IH}	Input High Voltage for PB0~PB1 Pins	5V	—	3.0	—	5.0	V
		—		0.6V_{DD}	—	V_{DD}	
R_{PH}	Pull-high Resistance for PB0~PB1 Pins	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I_{OL}	PB0/PB1 Sink Current	3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0000B	4.2	6.0	—	mA
		5V		10.5	15	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0001B	9.1	13.0	—	mA
		5V		20.3	29.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0010B	13.3	19.0	—	mA
		5V		29.4	42.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0011B	17.5	25.0	—	mA
		5V		38.5	55.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0100B	20.3	29.0	—	mA
		5V		44.1	63.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0101B	26.6	38.0	—	mA
		5V		56.7	81.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0110B	30.1	43.0	—	mA
		5V		64.4	92.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=0111B	33.1	48.0	—	mA
		5V		71.4	102.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1000B	40.6	58.0	—	mA
		5V		84	120	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1001B	46.9	67.0	—	mA
		5V		95.9	137.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1010B	53.2	76.0	—	mA
		5V		107.8	154.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1011B	58.1	83.0	—	mA
		5V		117.6	168.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1100B	72	90	—	mA
		5V		144	180	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OL}	PB0/PB1 Sink Current	3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1101B	87.2	109.0	—	mA
		5V		170.4	213.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1110B	116.8	146.0	—	mA
		5V		217.6	272.0	—	mA
		3V	V_{OL}=0.1V_{DD}; DRV_EN=1 PWDC[3:0]=1111B	144	180	—	mA
		5V		256.8	321.0	—	mA
I_{OH}	PB0/PB1 Source Current	3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0000B	-4.2	-6.0	—	mA
		5V		-11.2	-16.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0001B	-9.1	-13.0	—	mA
		5V		-21.7	-31.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0010B	-13.3	-19.0	—	mA
		5V		-31.5	-45	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0011B	-17.5	-25	—	mA
		5V		-40.6	-58.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0100B	-19.6	-28.0	—	mA
		5V		-46.2	-66.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0101B	-25.9	-37.0	—	mA
		5V		-59.5	-85	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0110B	-29.4	-42.0	—	mA
		5V		-67.2	-96.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=0111B	-32.9	-47.0	—	mA
		5V		-73.5	-105.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1000B	-39.2	-56.0	—	mA
		5V		-86.1	-123.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1001B	-44.8	-64.0	—	mA
		5V		-97.3	-139.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1010B	-50.4	-72.0	—	mA
		5V		-109.2	-156.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1011B	-55.3	-79.0	—	mA
		5V		-117.6	-168.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1100B	-68	-85	—	mA
		5V		-143.2	-179.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1101B	-81.6	-102.0	—	mA
		5V		-167.2	-209.0	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1110B	-106.4	-133.0	—	mA
		5V		-208	-260	—	mA
		3V	V_{OH}=0.9V_{DD}; DRV_EN=1 PWDC[3:0]=1111B	-128	-160	—	mA
		5V		-240	-300	—	mA

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



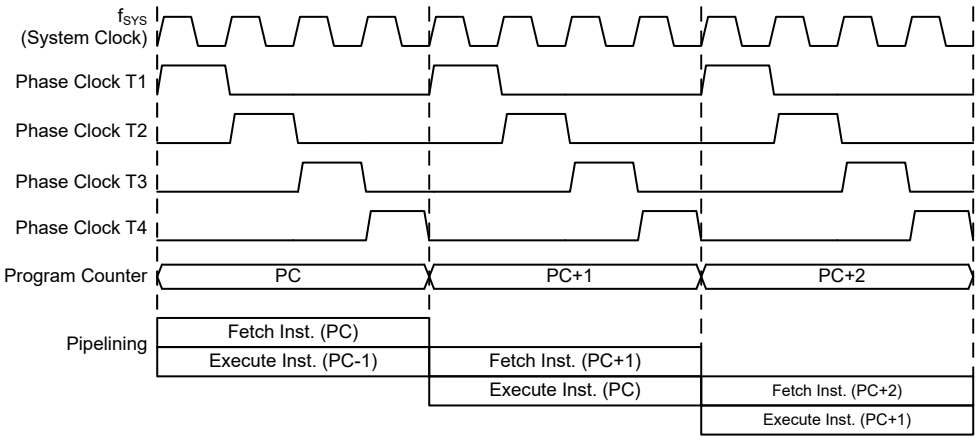
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The range of devices take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions 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.

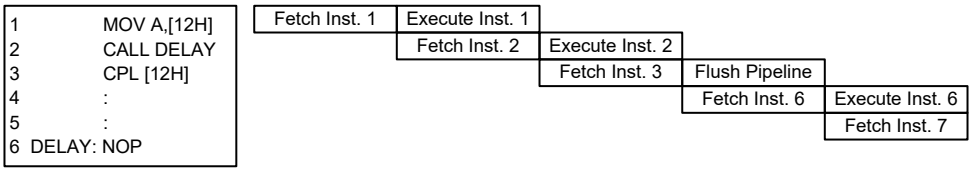
Clocking and Pipelining

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

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

Program Counter	
High Byte	**Low Byte (PCL)**
PC9~PC8	PCL7~PCL0

Program Counter

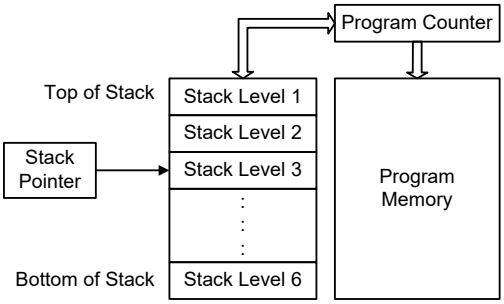
The lower byte of the Program Counter, known as the Program Counter Low Byte register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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 six levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



Arithmetic and Logic Unit – ALU

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

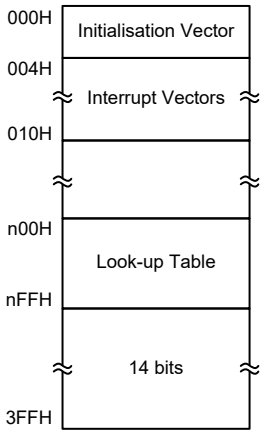
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement: INCA, INC, DECA, DEC
- Branch decision: JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET,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

Flash Program Memory

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

Structure

The Program Memory has a capacity of 1K×14 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

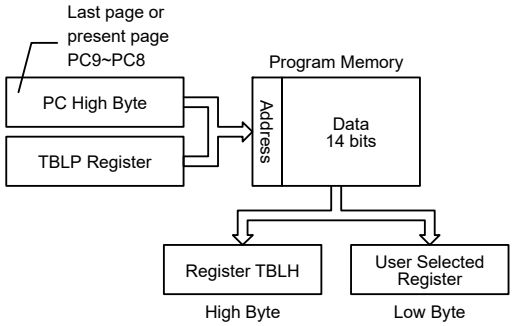


Table Program Example

The following example shows how the table pointer and table data are 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 "300H" which refers to the start address of the last page within the 1K Program Memory of the devices. 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 "306H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by TBLP if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

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

Table Read Program Example

```
tempreg1 db ?   ; temporary register #1
tempreg2 db ?   ; temporary register #2
:
:
mov a,06h       ; initialise low table pointer - note that this address is referenced
mov tblp,a      ; to the last page or present page
:
:
tabrd tempreg1  ; transfers value in table referenced by table pointer data at program
                ; memory address "306h" 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 "305h" transferred to tempreg2 and TBLH
                ; in this example the data "1AH" is transferred to tempreg1
                ; and data "0FH" to register tempreg2 and the data "00H" is transferred to
                ; TBLH
:
:
org 300h        ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

In Circuit Programming – ICP

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

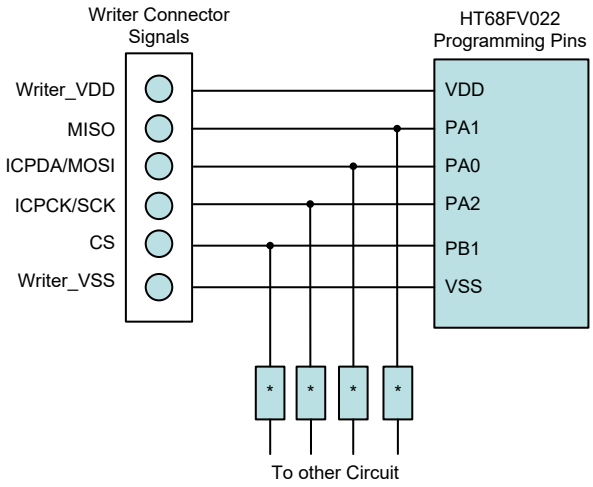
As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 6-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or 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 devices.

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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPCK/SCK	PA2	Programming Serial Data/Address
ICPDA/MOSI	PA0	Programming Clock
CS	PB1	Programming Chip Select
MISO	PA1	Programming Data
VDD	VDD	Power Supply
VSS	VSS	Ground

The Program Memory can be programmed serially in-circuit using this 6-wire interface. Data is downloaded and uploaded serially on two pins and an additional line for the clock and one pin for chip select. Two additional lines are required for the power supply. The technical details regarding the in-circuit programming of the devices are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process, the user must take control of the ICPDA/MOSI, ICPCK/SCK, MISO and CS pins for data and clock programming purposes to ensure that no other outputs are connected to these four pins.



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

On Chip Debug Support – OCDS

There is an EV chip named HT68VV022/V024 which is used to emulate the HT68FV022/V024 device. This 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 and package type. 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. 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	Power Supply
VSS	VSS	Ground

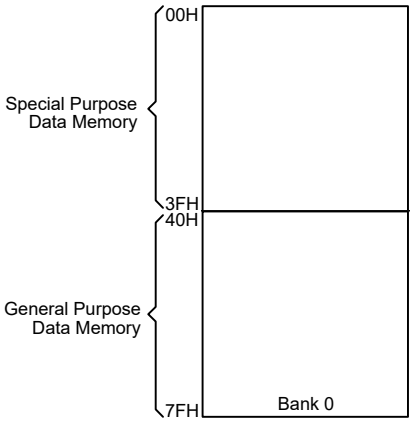
Data Memory

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

Structure

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

The start address of the Data Memory for the devices is 00H. The address range of the Special Purpose Data Memory for the devices is from 00H to 3FH while the General Purpose Data Memory address range is from 40H to 7FH.



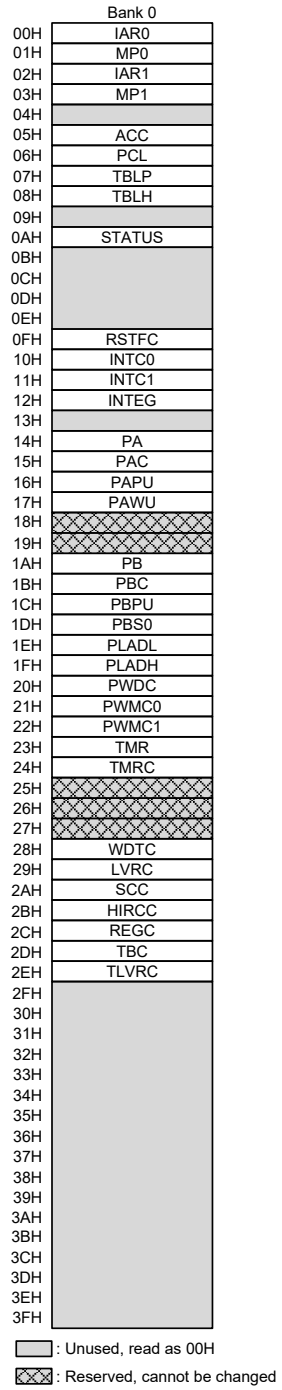
Data Memory Structure

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”.



Special Purpose Data Memory Structure

Voice Flash Memory

The devices include a fully integrated 16Mbit~32Mbit Voice Flash Memory to store voice data, which can be edited using Holtek Voice MCU Workshop. After edited in the IDE3000 using Holtek voice library, it can then be programmed using the HOPE3000 or e-Writer32. For more details about how to use the voice libraries, refer to the related documentation.

The Voice Flash Memory is powered by an internal 3.0V LDO whose power supply ranges from 2.3V to 3.6V. When the V_{DD} is greater than 3.6V, the LDO should be set to "on". For more control methods, refer to the Voltage Regulator section.

Device	Capacity
HT68FV022	16Mbit
HT68FV024	32Mbit

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0/IAR1 and MP0/MP1 can together access data from Bank 0. As the Indirect Addressing Register is not physically implemented, reading the Indirect Addressing Register will return a result of "00H" and writing to the register will result in no operation.

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1, are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the Memory Pointer. MP0/MP1, together with Indirect Addressing Register, IAR0/IAR1, are used to access data from Bank 0.

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

Indirect Addressing Program

```
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:
```

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

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, TBLH

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

Status Register – STATUS

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

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

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

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

In addition, on entering an interrupt sequence or executing a subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status register 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 instructions, the CZ flag is equal to the Z flag.
For SBC/SBCM 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.

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 operations are selected through the combination of configuration option and relevant control registers.

Oscillator Overview

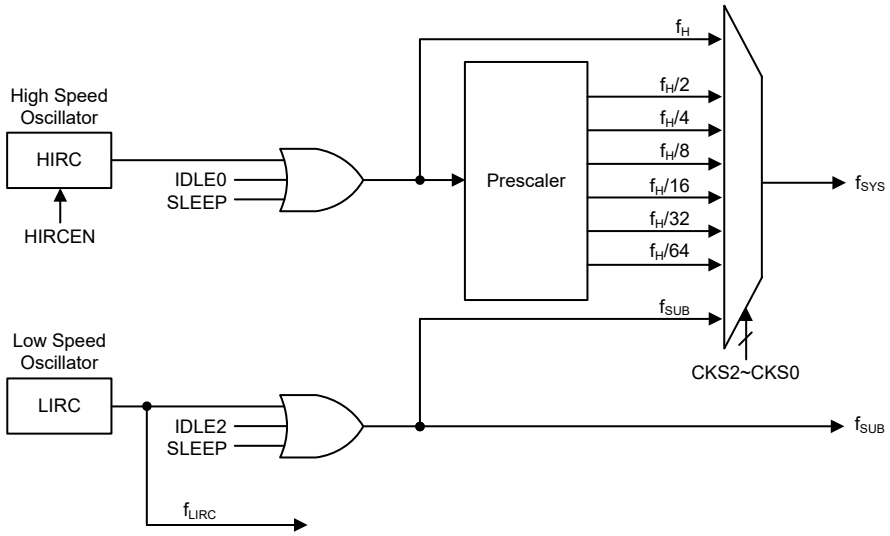
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupt. Two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. 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	8/12/16MHz
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 sourced from the internal 8/12/16MHz 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.



System Clock Configurations

Internal High Speed RC Oscillator – HIRC

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

Internal 32kHz Oscillator – LIRC

The internal 32kHz System Oscillator is also 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.

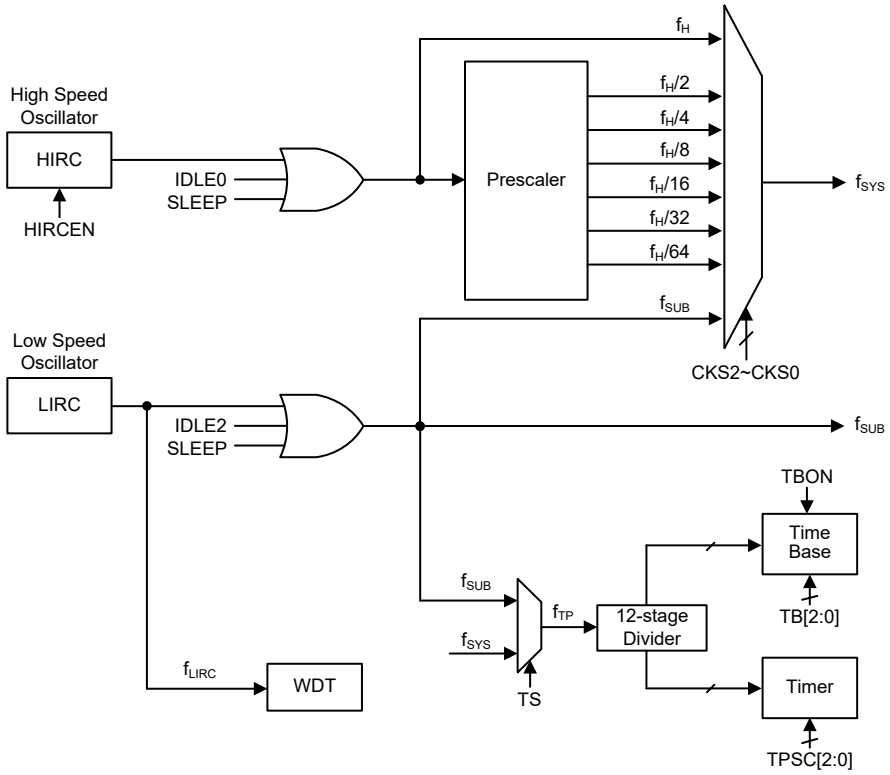
Operating Modes and System Clocks

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 using register programming, a clock system can be configured to obtain maximum application performance.

The main system clock, can come from a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source is 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 FAST Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Modes are used when the microcontroller CPU is switched off to conserve power.

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

"x": Don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.

2. The f_{LIRC} clock will be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

FAST Mode

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

SLOW Mode

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

SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bit both are low. In the SLEEP mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral function will also be stopped. However, the f_{LIRC} clock can continue 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 SCC and HIRCC registers are used to control the system clock and the HIRC oscillator configurations.

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

System Operating Mode Control Register List

• SCC Register

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	0	0	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. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.

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

• **HIRCC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection
00: 8MHz
01: 12MHz
10: 16MHz
11: 8MHz
When the HIRC oscillator is enabled or the HIRC frequency selection is changed by application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1. It is recommended that the HIRC frequency selected by these bits should be the same as the frequency determined by the configuration option to keep the HIRC frequency accuracy specified in the A.C. Characteristics.

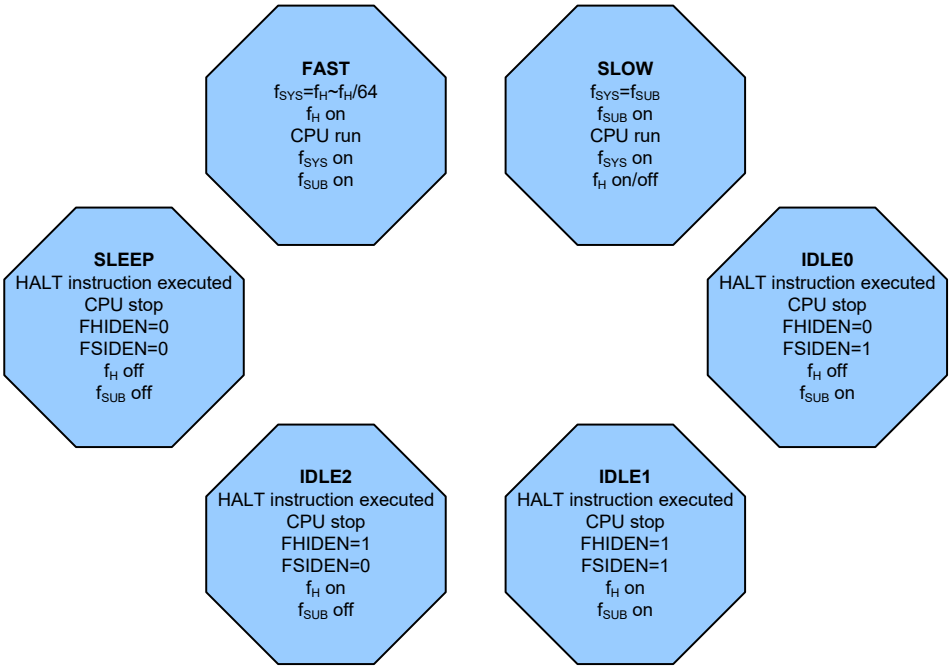
Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is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

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

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.

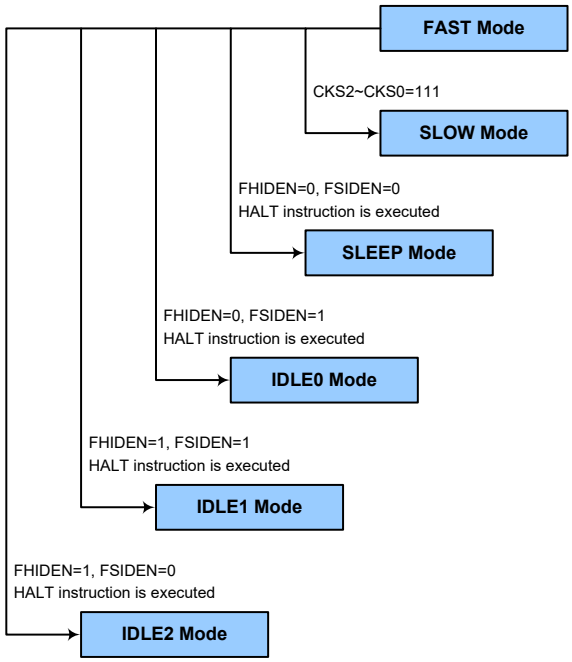
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the devices enter the IDLE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



FAST Mode to SLOW Mode Switching

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

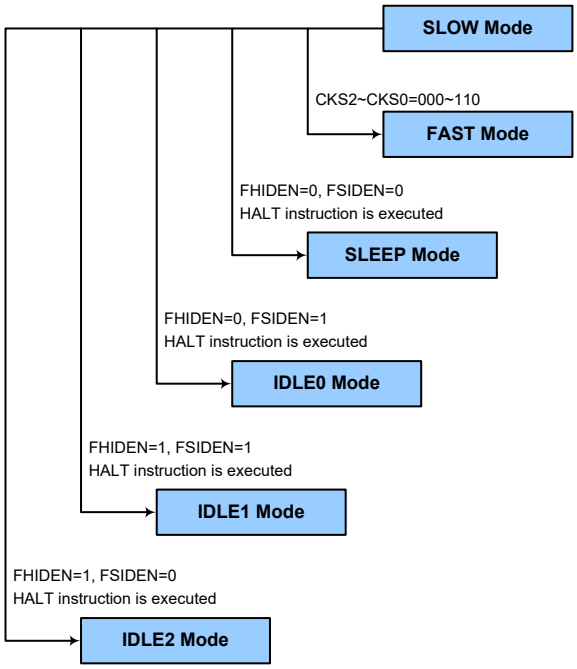
The SLOW Mode system clock is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

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

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



Entering the SLEEP Mode

There is only one way for the devices 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.

Entering the IDLE0 Mode

There is only one way for the devices 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.

Entering the IDLE1 Mode

There is only one way for the devices 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 devices 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 devices to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Modes, 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 devices. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the devices which have different package types, as there may be unbonded pins. These pins 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 Modes 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 devices are woken up again, they will take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

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

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

When the devices execute the "HALT" instruction, they will enter the IDLE or SLEEP mode and the PDF flag will be set high. The PDF flag is cleared to 0 if the devices experience a system power-up or executs the clear Watchdog Timer instruction.

If the system is woken up by a WDT overflow, a Watchdog Timer time-out reset will be initiated and the TO flag will be set to 1. The TO flag is set high if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake up the system. When a 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 devices will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{LIRC} which is sourced from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of $[(2^8\text{-}2^0)\sim2^8]$ to $[(2^{15}\text{-}2^7)\sim2^{15}]$ to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as Watchdog Timer the enable/disable and the MCU reset operation.

• **WDTC Register**

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

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

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $[(2^8\text{-}2^0)\sim2^8]/f_{LIRC}$
001: $[(2^9\text{-}2^1)\sim2^9]/f_{LIRC}$
010: $[(2^{10}\text{-}2^2)\sim2^{10}]/f_{LIRC}$
011: $[(2^{11}\text{-}2^3)\sim2^{11}]/f_{LIRC}$
100: $[(2^{12}\text{-}2^4)\sim2^{12}]/f_{LIRC}$
101: $[(2^{13}\text{-}2^5)\sim2^{13}]/f_{LIRC}$
110: $[(2^{14}\text{-}2^6)\sim2^{14}]/f_{LIRC}$
111: $[(2^{15}\text{-}2^7)\sim2^{15}]/f_{LIRC}$
These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period.

• **RSTFC Register**

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

"x": unknown

Bit 7~3 Unimplemented, read as "0"

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

Watchdog Timer Operation

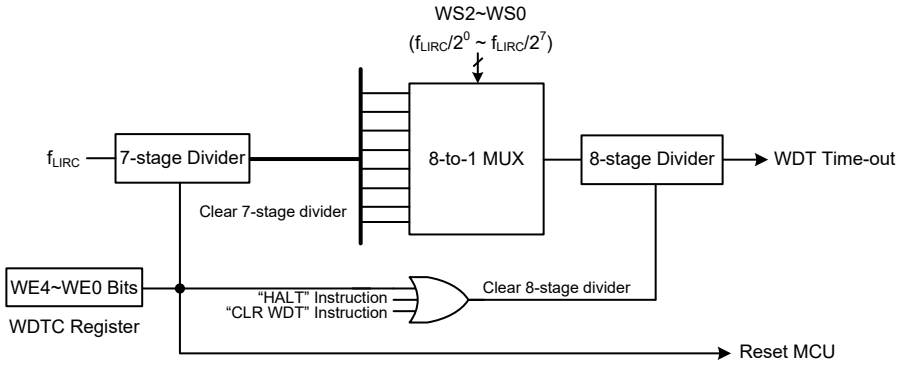
The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using 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 devices. There are five bits, WE4~WE0, in the WDTC register to offer the Watchdog Timer enable/disable control and the MCU reset. 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 devices after a delay time, t_{SRESET}. After power on these bits will have a value of 01010B.

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

Watchdog Timer Enable/Disable Control

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

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



Watchdog Timer

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the devices can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power that 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.

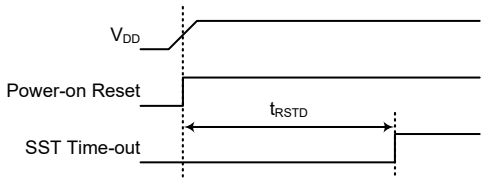
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

Reset Functions

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

Power-on Reset

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

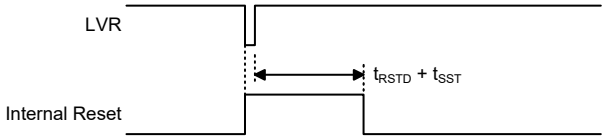


Power-on Reset Timing Chart

Low Voltage Reset – LVR

The microcontrollers contain a low voltage reset circuit in order to monitor the supply voltage of the devices and provide an MCU reset when the value falls below a certain predefined level. If the supply voltage of the devices drop 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 devices 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 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 t_{LVR} value can be selected by the TLVR1~TLVR0 bits in the TLVRC register. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by environmental noise, the LVR will reset the devices after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set high. After power on the LVRC register will have the value of 01100110B.

Note that the LVR function will be automatically disabled when the devices enter the SLEEP or IDLE mode.



Low Voltage Reset Timing Chart

Low Voltage Reset Registers

The LVRC and TLVRC registers are used to control the Low Voltage Reset function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
LVRC	LVS7	LVS6	LVS5	LVS4	LVS3	LVS2	LVS1	LVS0
TLVRC	—	—	—	—	—	—	TLVR1	TLVR0

Low Voltage Reset Register List

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select
01100110: 1.7V
01010101: 2.2V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
11110000: LVR disable
Other values: Generates a MCU reset – register is reset to POR value

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

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

- **TLVRC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TLVR1	TLVR0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	1

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **TLVR1~TLVR0**: Minimum low voltage width to reset time (t_{LVR}) selection
00: (7~8)×t_{LIRC}
01: (31~32)×t_{LIRC}
10: (63~64)×t_{LIRC}
11: (127~128)×t_{LIRC}

• **RSTFC Register**

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

"x": unknown

Bit 7~3 Unimplemented, read as "0"

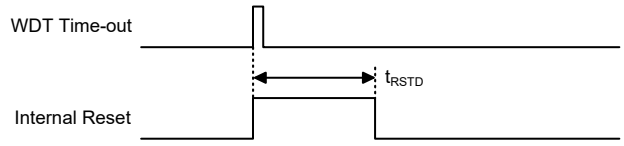
Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set high when an actual Low Voltage Reset situation occurs. This bit can only be cleared to zero by the application program.

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

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

Watchdog Time-out Reset during Normal Operation

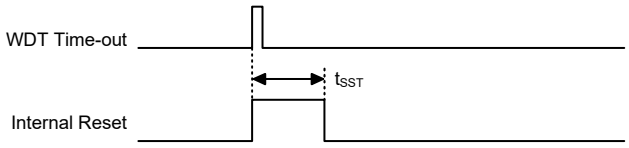
When the Watchdog Time-out Reset during normal operation in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

The Watchdog Time-out Reset during SLEEP or IDLE Mode is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to "0" and the TO 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	LVR reset during FAST or SLOW Mode operation
1	u	WDT time-out reset during FAST or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

"u" stands for unchanged

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

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

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

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	xxxx xxxx	uuuu uuuu	uuuu uuuu
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu
IAR1	xxxx xxxx	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	-000 -000	-000 -000	-uuu -uuu
INTEG	---- --00	---- --00	---- --uu
PA	1111 -111	1111 -111	uuuu -uuu
PAC	1111 -111	1111 -111	uuuu -uuu
PAPU	0000 -000	0000 -000	uuuu -uuu
PAWU	0000 -000	0000 -000	uuuu -uuu
PB	---- --11	---- --11	---- --uu
PBC	---- --11	---- --11	---- --uu
PBPU	---- --00	---- --00	---- --uu
PBS0	---- 0000	---- 0000	---- uuuu

Register	Power-On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PLADL	0000 0000	0000 0000	uuuu uuuu
PLADH	1000 0000	1000 0000	1uuu uuuu
PWDC	---- 0100	---- 0100	---- uuuu
PWMC0	000-0000	000- 0000	uuu- uuuu
PWMC1	000- 0000	000- 0000	uuu- uuuu
TMR	0000 0000	0000 0000	uuuu uuuu
TMRC	0000 1000	0000 1000	uuuu uuuu
WDTC	0101 0111	0101 0111	uuuu uuuu
LVRC	0110 0110	0110 0110	uuuu uuuu
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- 0001	---- 0001	---- uuuu
REGC	---- --00	---- --00	---- --uu
TBC	0--- -000	0--- -000	u--- -uuu
TLVRC	---- --01	---- --01	---- --uu

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~PB. 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	D7	D6	D5	D4	—	PA2	PA1	PA0
PAC	D7	D6	D5	D4	—	PAC2	PAC1	PAC0
PAPU	D7	D6	D5	D4	—	PAPU2	PAPU1	PAPU0
PAWU	D7	D6	D5	D4	—	PAWU2	PAWU1	PAWU0
PB	—	—	—	—	—	—	PB1	PB0
PBC	—	—	—	—	—	—	PBC1	PBC0
PBPU	—	—	—	—	—	—	PBPU1	PBPU0

“—”: Unimplemented, read as “0”

I/O Logic Function Register List

Pull-high Resistors

Many product applications require pull-high resistors or pull-low 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 or pull-low resistor. These pull-high resistors are selected using the PxPU register and are implemented using weak PMOS transistors.

Note that the pull-high resistors 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. However, the actual available bits for each I/O Port may be different.

Note that the control bits denoted as "Dn" in the PAPU register should be kept unchanged after power-on reset.

Port A Wake-up

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

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

• PAWU Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	—	PAWU2	PAWU1	PAWU0
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 7~4 **D7~D4**: Reserved, cannot be changed

Bit 3 Unimplemented, read as "0"

Bit 2~0 **PAWU2~PAWU0**: PA2~PA0 wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

Each I/O port has its own control register known as PAC~PBC, 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. However, the actual available bits for each I/O Port may be different.

Note that the control bits denoted as "Dn" in the PAC register should be cleared to 0 to set the corresponding pin as an output after power-on reset. This can prevent the devices from consuming power due to input floating states for any unbonded pins.

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

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

• **PBS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PBS03	PBS02	PBS01	PBS00
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

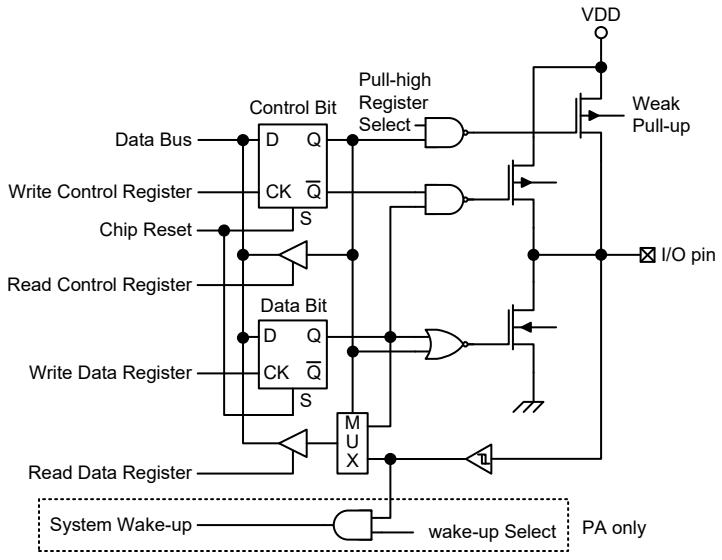
Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **PBS03~PBS02**: PB1 Pin-shared function selection
00: PB1
01: PB1
10: PB1
11: PWM2

Bit 1~0 **PBS01~PBS00**: PB0 Pin-shared function selection
00: PB0
01: PB0
10: PB0
11: PWM1

I/O Pin Structures

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



Logic Function Input/Output Structure

Programming Considerations

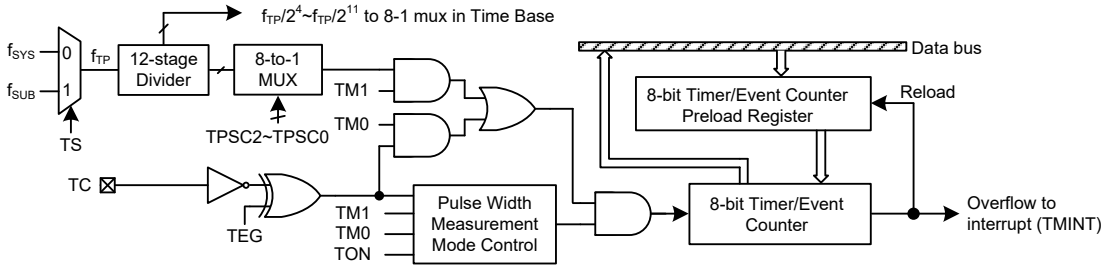
Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high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

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

Timer/Event Counter

The provision of the Timer/Event Counter forms an important part of any microcontroller, giving the designer a means of carrying out time related functions. The devices contain an 8-bit Timer/Event Counter, which contains an 8-bit programmable count-up counter and the clock may come from an external or internal clock source. As the timer has three different operating modes, it can be configured to operate as a general timer, an external event counter or a pulse width measurement device.



8-bit Timer/Event Counter

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 TS bit in the TMRC Timer Control Register to be derived from the f_{SYS} or f_{SUB} clock, the division ratio of which is selected by the TPSC2~TPSC0 bits in the TMRC Timer/Event Control Register.

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 TC pin. Depending upon the condition of the TEG 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 registers related to the Timer/Event Counter. The first is the TMR register that contains the actual value of the timer and into which an initial value can be preloaded. Writing to the TMR register will transfer the specified data to the Timer/Event Counter. Reading the TMR register will read the contents of the Timer/Event Counter. The second is the TMRC control register, which is used to define the operating mode, select the internal clock source, control the counting enable or disable and select the active edge.

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMR	D7	D6	D5	D4	D3	D2	D1	D0
TMRC	TM1	TM0	TS	TON	TEG	TPSC2	TPSC1	TPSC0

Timer/Event Counter Register List

Timer Register – TMR

The timer register TMR is the place where the actual timer value is stored. The value in the timer register 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 FFH for the 8-bit Timer/Event Counter, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be loaded with the preload register value and continue counting.

Note that to achieve a maximum full range count of FFH, the preload register must first be cleared. Note that 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 remain in the preload register and will only be written into the actual counter until an overflow occurs.

• **TMR 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 byte

Timer Control Register – TMRC

The flexible features of the Holtek microcontroller Timer/Event Counter are implemented by operating in three different modes, the options of which are determined by the contents of control register bits.

The Timer Control Register is known as TMRC. It is the Timer Control Register together with its corresponding timer register 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 TM1~TM0 bits in the Timer Control Register must be set to the required logic levels. The timer-on bit TON 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 f_{SYS} or f_{SUB} clock selected by setting the TS bit. Bits TPSC2~TPSC0 determine the division ratio of the selected clock source. The internal clock selection will 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 type is selected by the logic level of the TEG bit in the Timer Control Register.

• **TMRC Register**

Bit	7	6	5	4	3	2	1	0
Name	TM1	TM0	TS	TON	TEG	TPSC2	TPSC1	TPSC0
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 **TM1~TM0**: Timer/Event Counter operating mode selection
00: Unused
01: Event Counter Mode
10: Timer Mode
11: Pulse Width Measurement Mode

Bit 5 **TS**: Timer f_{TP} clock source selection
0: f_{SYS}
1: f_{SUB}

Bit 4 **TON**: Timer/Event Counter counting enable
0: Disable
1: Enable

Bit 3 **TEG**: Timer/Event Counter 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 **TPSC2~TPSC0**: Timer internal clock selection
000: f_{TP}
001: $f_{TP}/2$
010: $f_{TP}/4$
011: $f_{TP}/8$
100: $f_{TP}/16$
101: $f_{TP}/32$
110: $f_{TP}/64$
111: $f_{TP}/128$

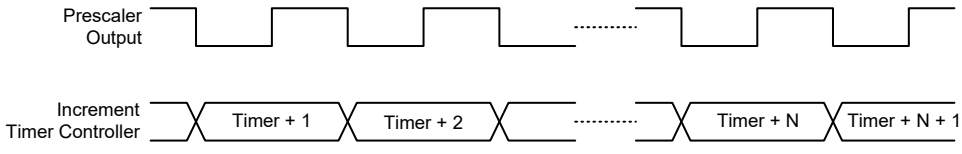
Timer/Event Counter Operating Modes

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 TM1 and TM0 bits in the TMRC register.

Timer Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "10" respectively. In this mode, the Timer/Event Counter can be utilised 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 f_{TP} is used as the timer clock, which can be selected to be devirved from f_{SYS} or f_{SUB} by setting the TS bit in the TMRC register. The division of the f_{TP} clock is selected by the TPSC2~TPSC0 bits in the same register. The timer-on bit TON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increases by one. When the timer reaches its maximum 8-bit, FFH Hex, value and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. It should be noted that in the Timer mode, even if the devices are in the IDLE/SLEEP mode, if the selected internal clock is still activated and a timer overflow occurs, it will generate a timer interrupt and corresponding wake-up source.



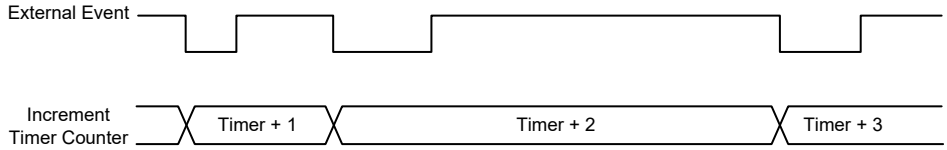
Timer Mode Timing Chart

Event Counter Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "01" respectively. In this mode, a number of externally changing logic events, occurring on the external timer TC pin, can be recorded by the Timer/Event Counter.

When operating in this mode, the external timer pin, TC, is used as the Timer/Event Counter clock source. After the other bits in the Timer Control Register have been properly configured, the enable bit TON, can be set high to enable the Timer/Event Counter. If the Active Edge Selection bit, TEG, is low, the Timer/Event Counter will increase each time the TC pin receives a low to high transition. If the TEG bit is high, the counter will increase each time the TC pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting.

As the exernal pin TC is pin-shared with PA1, the corresponding bit in the I/O port control register must be set high to set the pin as the input. It should be noted that in the Event Counter mode, even if the devices are in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (TEG=1)

Pulse Width Measurement Mode

To select this mode, bits TM1 and TM0 in the TMRC 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.

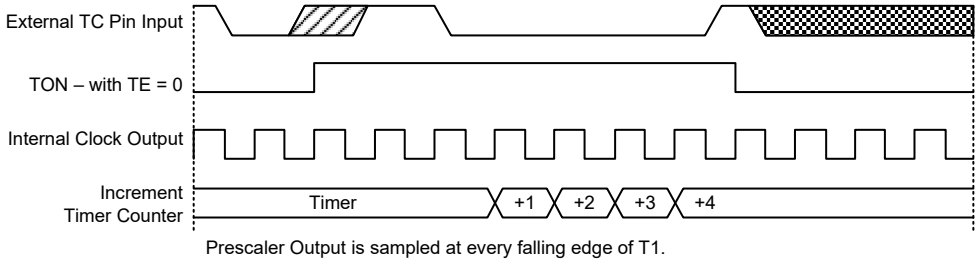
When operating in this mode the internal clock f_{TP} is used as the timer clock, which can be selected to be devirved from f_{SYS} or f_{SUB} by setting the TS bit in the TMRC register. The division of the f_{TP} clock is selected by the TPSC2~TPSC0 bits in the same register. After the other bits in the Timer Control Register have been properly configured, the enable bit TON, 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 TC pin.

If the active Edge Selection bit TEG is low, once a high to low transition has been received on the TC pin, the Timer/Event Counter will start counting based on the selected internal clock source until the TC 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 Selection bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the TC pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will then be automatically reset to zero. 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 TC 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 TC 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 measurements 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 signal 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 devices are in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC pin if the selected internal clock source is still activated and the external signal continues to change state. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Pulse Width Measurement Mode Timing Chart (TEG=0)

Programming Considerations

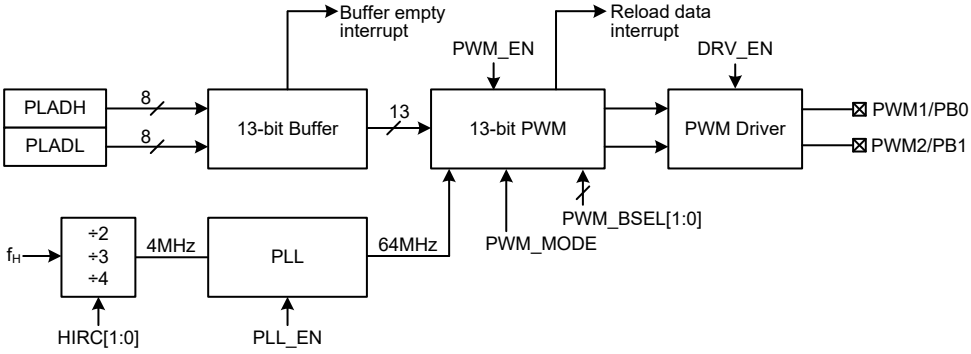
When running in the Timer Mode, if the internal system clock is used as the timer clock source, the timer can be synchronised with the overall operation of the microcontroller. In this mode when the timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector. For the Pulse Width Measurment Mode, the internal 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 errors 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 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 bit 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 issued to ensure the timer is properly configured for the required applications. It is also important to ensure that a desired initial value is first loaded into the timer register 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 to generate an interrupt signal. If the Timer/Event Counter interrupt is enabled this will in turn allow program branch to its interrupt vector. However irrespective of whether the interrupt is enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the devices are in the IDLE/SLEEP mode. This situation may occur if the Timer/Event Counter internal clock source is still activated 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 devices 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.

Audio PWM Driver

The devices contain a fully integrated PWM driver, which completes with volume control and mute function. The voice data located in the PLADH and PLADL register pair can be output to the external speaker via the PWM1 and PWM2 pins. The PWM driver offers the possibility of directly driving external speakers, and supports up to 13-bit Green mode and up to 12-bit Normal mode PWM output. The volume control can be adjusted using the PWDC3~PWDC0 bits.



Audio PWM Driver Block Diagram

Audio PWM Driver Registers

The overall voice play function is controlled using a series of registers. Three control registers exist to control the PWM function together with volume control and mute function. Two data registers exist to store the data which is to be played.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PWDC	—	—	—	—	PWDC3	PWDC2	PWDC1	PWDC0
PWMC0	PWM_MODE	PWM_BSEL1	PWM_BSEL0	—	MUTEB	PLL_EN	DRV_EN	PWM_EN
PWMC1	BUFCLR	BUFFLAG1	BUFFLAG0	—	NORMALC	PWPRCN	INSERT1	INSERT0
PLADL	P_D7	P_D6	P_D5	P_D4	P_D3	P_D2	P_D1	P_D0
PLADH	P_D15	P_D14	P_D13	P_D12	P_D11	P_D10	P_D9	P_D8

Audio PWM Driver Register List

- **PWDC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PWDC3	PWDC2	PWDC1	PWDC0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	1	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **PWDC3~PWDC0**: PWM1/PWM2 source/sink current control
0000: Source/Sink current=Level 0
0001: Source/Sink current=Level 1
0010: Source/Sink current=Level 2
0011: Source/Sink current=Level 3
0100: Source/Sink current=Level 4
0101: Source/Sink current=Level 5
0110: Source/Sink current=Level 6
0111: Source/Sink current=Level 7
1000: Source/Sink current=Level 8

1001: Source/Sink current=Level 9
1010: Source/Sink current=Level 10
1011: Source/Sink current=Level 11
1100: Source/Sink current=Level 12
1101: Source/Sink current=Level 13
1110: Source/Sink current=Level 14
1111: Source/Sink current=Level 15

Note: Refer to the PWM/PB0/PB1 Driver Characteristics section to obtain the exact value.

• **PWMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PWM_MODE	PWM_BSEL1	PWM_BSEL0	—	MUTEB	PLL_EN	DRV_EN	PWM_EN
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 **PWM_MODE**: PWM Mode control
0: Normal Mode – PWM signal will drive both PWM1 and PWM2 simultaneously
1: Green Mode – PWM signal will drive PWM1 and PWM2 alternately
The PWM1 and PWM2 pin functions should be properly configured using the corresponding pin-shared control bits depending on the selected PWM mode.

Bit 6~5 **PWM_BSEL1~PWM_BSEL0**: PWM effective bit number selection
00: 9-bit
01: 10-bit
10: 11-bit
11: 12-bit
These bits are used to select the effective bits in the PLADH and PLADL registers, counting from the highest bit P_D15. The effective bits of PWM in the Green mode need add a sign bit, which is PWM_BSEL[1:0]+1.

Bit 4 Unimplemented, read as "0"

Bit 3 **MUTEB**: PWM mute control
0: Mute PWM output
1: Enable PWM output
This bit is used to enable the PWM output function. When this bit is cleared to 0, the PWM output will be muted and the PLADH/PLADL will be the default value.

Bit 2 **PLL_EN**: PLL control
0: Disable
1: Enable
When the PLL generator is disabled by clearing this bit, the PLL output will be low.

Bit 1 **DRV_EN**: PWM driver control
0: Disable
1: Enable
If the PWM driver is disabled, both of the PWM1 and PWM2 outputs will be floating.

Bit 0 **PWMEN**: PWM function control
0: Disable
1: Enable
If the PWM function is disabled, the PWM output is low level.

Note: 1. When the devices execute the "HALT" instruction, the PWM, driver and PLL will be disabled by hardware.
2. When MUTEB=0, PWM_EN=1 and DRV_EN=1, the PWM output digital value is 8000H.
3. When the PWM_EN is from 0 to 1, the first value of the PWM is 8000H and the second value is the PLADH/PLADL current value.

• **PWMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	BUFCLR	BUFFLAG1	BUFFLAG0	—	NORMALC	PWPRCN	INSERT1	INSERT0
R/W	R/W	R	R	—	R/W	R/W	R/W	R/W
POR	0	0	0	—	0	0	0	0

Bit 7 **BUFCLR**: Set buffer to default value of PLADH/PLADL
0→1→0: Set buffer to default value of PLADH/PLADL

Bit 6~5 **BUFFLAG1~BUFFLAG0**: Buffer status flag (only write by hardware)
00: Both buffer and PLADH/PLADL have no data
01: Buffer has written data, PLADH/PLADL has no data
10: Both buffer and PLADH/PLADL have written data
11: Unknown status, clear buffer (refer to the BUFCLR bit in the PWMC1 register)

When the buffer is empty or read by hardware, the PLADH/PLADL value will be automatically loaded to the buffer. Note that it should first write the PLADL value to the buffer, then write the PLADH value to the buffer.

One PWM clock before the end of the current PWM period, the buffer is read by hardware, and the RELDF flag occurs. The buffer and PLADH/PLADL are empty if no data is written to the PLADH/PLADL, the EMPTF flag occurs. The buffer and PLADH/PLADL are not empty if writting data to the PLADH/PLADL, the EMPTF flag does not occur.

Just only for writes DATA to PLADH/PLADL will BUFFLAG[1:0]+1.

Bit 4 Unimplemented, read as "0"

Bit 3 **NORMALC**: Normal mode operation selection
0: Symmetry operation
1: Asymmetric operation

Bit 2 **PWPRCN**: Number of PWM period control
0: 1 period
1: 2 periods

Bit 1~0 **INSERT1~INSERT0**: Selection of insertion point number
00: 0
01: 1
10: 3
11: 7

• **PLADL Register**

Bit	7	6	5	4	3	2	1	0
Name	P_D7	P_D6	P_D5	P_D4	P_D3	P_D2	P_D1	P_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 **P_D7~P_D0**: Play data low byte register bit 7 ~ bit 0

This register is used to store the PWM play data low byte. Note that before writing data to this register, the PWM function should be enabled first and the low byte play data register should be modified followed by the high byte play data register being written if the PWM play data is necessary to be updated. If the play data has N effective bits determined by the PWM_BSEL[1:0] bits, then the effective bits are from P_D15 to P_Dx, where x is equal to 16-N-1 for Green mode and is equal to 16-N for Normal mode.

• **PLADH Register**

Bit	7	6	5	4	3	2	1	0
Name	P_D15	P_D14	P_D13	P_D12	P_D11	P_D10	P_D9	P_D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	0	0	0	0	0

Bit 7~0 **P_D15~P_D8**: Play data high byte register bit 7 ~ bit 0

This register is used to store the PWM play data high byte. Note that before writing data to this register, the PWM function should be enabled first and the low byte play data register should be modified followed by the high byte play data register being written if the PWM play data is necessary to be updated. If the play data has N effective bits determined by the PWM_BSEL[1:0] bits, then the effective bits are from P_D15 to P_Dx, where x is equal to 16-N-1 for Green mode and is equal to 16-N for Normal mode.

Note: 1. When MUTEB=0, the PLADH/PLADL will always be 8000H and cannot be written.
2. When PLADH/PLADL=0000H or 0001H, the PWM output will be the signal of PLADH/PLADL=0000H.

Audio PWM Driver Operation

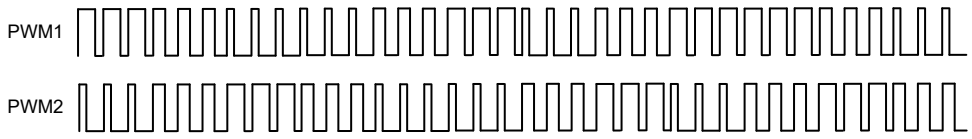
The PWM has two output modes, Normal mode and Green mode. The waveform is shown in the figure below. The main differences are as follows:

(1) One of the PWM1 and PWM2 has signal change in the Green mode, but the PWM1 and PWM2 both have the signal change in the Normal mode.

(2) When digital code in the middle value: The output of PWM1 and PWM2 are 0 in the Green mode, but the output of PWM1 and PWM2 will change in the Normal mode.

Note: Care must be taken when Voice Data = 0/1/(2^n-1)/(2^n-2), this may result in unpredictable situations. Therefore, it is not recommended for use.

Green PWM Mode



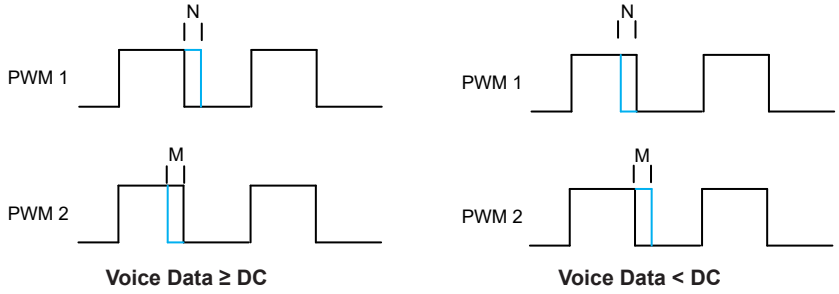
Normal PWM Mode

In the Normal mode, there are two operation modes, symmetry operation and asymmetric operation. The description are shown below: (Voice Data: the value obtained by the buffer; DC: the default value is PLADH/PLADL=8000H)

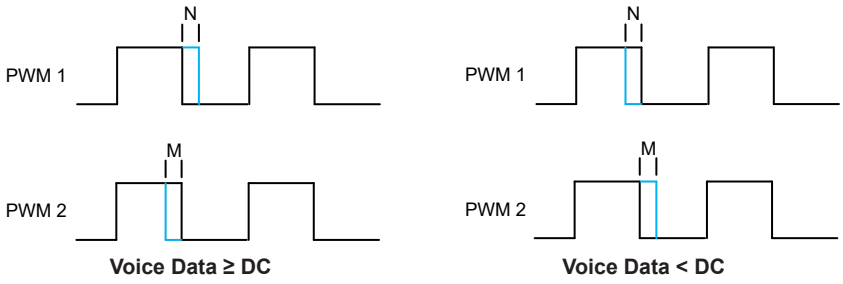
- Symmetry Operation (NORMALC=0)
  - Voice Data ≥ DC
    N=(Voice Data-DC)
    M=(DC-(Complement value of the Voice Data))
    PWM1=DC+N
    PWM2=DC-M

- Voice Data < DC
  N=(DC-Voice Data)
  M=((Complement value of the Voice Data)-DC)
  PWM1=DC-N
  PWM2=DC+M

The complement value of the Voice Data:
PWM 9-bit: 1FFH-Voice Data
PWM 10-bit: 3FFH-Voice Data
PWM 11-bit: 7FFH-Voice Data
PWM 12-bit: FFFH-Voice Data



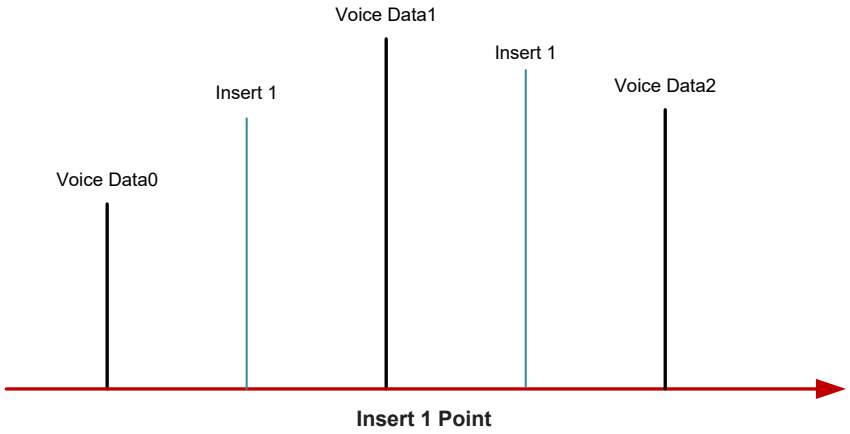
- Asymmetric Operation (NORMALC=1)
  - Voice Data ≥ DC
    N=(Voice Data-DC+1)
    M=(Voice Data-DC)
    PWM1=DC+N
    PWM2=DC-M
  - Voice Data < DC
    N=(DC-Voice Data)
    M=(DC-Voice Data+1)
    PWM1=DC-N
    PWM2=DC+M



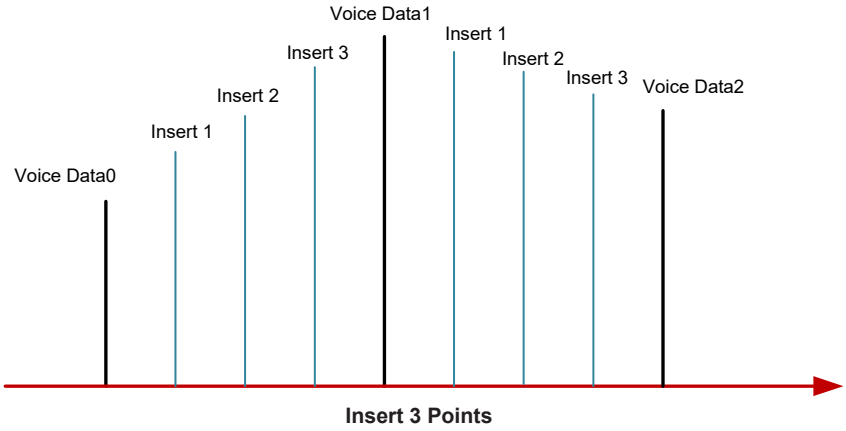
Using linear insertion point. The following example illustrates how to use insertion point: (Voice Data 1 and Voice Data 0: two consecutive values obtained by buffer).

- Judge which is greater, Voice Data 0 or Voice Data 1 (Assume Voice Data 1 is greater)
- x=(Voice Data1- Voice Data0)/(n+1) (n: insertion point number)
- Insert 1=Voice Data0+x
- Insert 2=Voice Data0+2x
  ⋮
  ⋮
- Insert 7=Voice Data0+7x

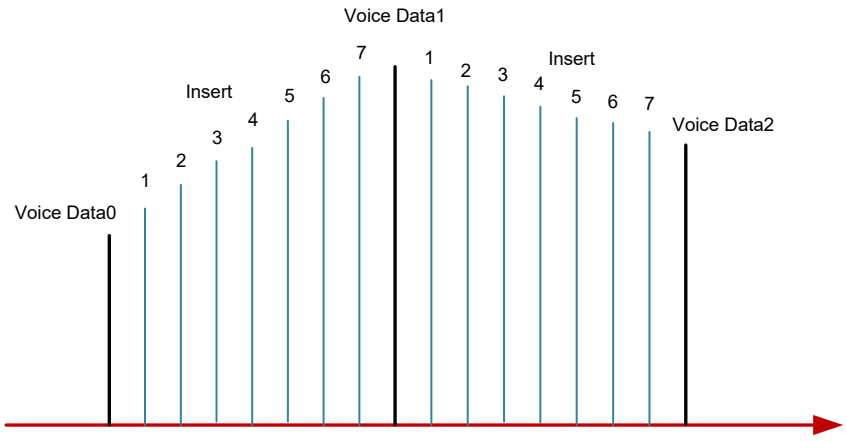
The diagram of insertion point is shown below.



Insert 1 Point



Insert 3 Points



Insert 7 Points

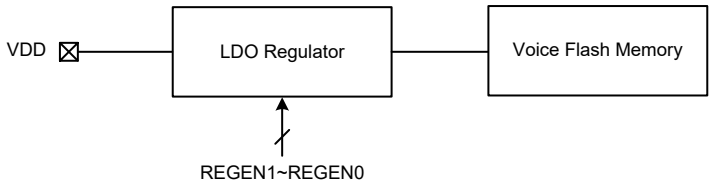
Audio PWM Driver Interrupts

The PWM has two internal interrupts, which are the reload data interrupt (RELOADINT) and the buffer empty interrupt (EMPTYINT). When the hardware has completed the PLADH/PLADL reload, the reload data interrupt request flag in the corresponding interrupt control register will be set. If the interrupt is enabled and the stack is not full, a reload data interrupt will be generated. When the buffer is empty, the buffer empty interrupt request flag in the corresponding interrupt control register will be set. If the interrupt is enabled and the stack is not full, a buffer empty interrupt will be generated. The corresponding interrupt signal will direct the program flow to the associated interrupt address for processing.

Voltage Regulator – LDO

The devices include a voltage regulator, LDO, which provides power to Voice Flash Memory. The REGC register controls the regulator module in three modes. In the disable mode, the LDO will be switched off and the Voice Flash Memory will be not powered. In the second mode the regulator is turned on, when the input voltage exceeds 3.0V, the LDO will output a fixed voltage of 3.0V to provide power the Voice Flash Memory. In the bypass mode, the LDO will also be switched off but the VDD will be directly passed to provide power the Voice Flash Memory.

Note that the maximum operating voltage for the internal Voice Flash Memory is 3.6V. When the VDD is greater than or equal to 3.6V, the LDO should be enabled to avoid damaging the internal Voice Flash Memory.



LDO Regulator Block Diagram

- **REGC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	REGEN1	REGEN0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **REGEN1~REGEN0**: Regulator of/off control
00: Regulator off, disable mode, the LDO output is pulled low
01: Regulator on, the Voice Flash Memory power is equal to 3.0V
10: Regulator on, the Voice Flash Memory power is equal to 3.0V
11: Regulator off, bypass mode, the Voice Flash Memory power is equal to V_{DD}

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter or Time Base 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 an external interrupt and several internal interrupt functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions including the Timer/Event Counter, PWM and Time Base.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory. The registers fall into two categories. The first is the INTC0~INTC1 registers which setup the primary interrupts, the second is the INTEG register which sets 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
Global		EMI	—
INT pin		INTE	INTF
Timer/Event Counter		TE	TF
Time Base		TBE	TBF
PWM	Buffer empty interrupt	EMPTE	EMPTF
	Reload data interrupt	RELDE	RELDF

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	TF	D5	INTF	TE	D2	INTE	EMI
INTC1	—	RELDF	EMPTF	TBF	—	RELDE	EMPTE	TBE

Interrupt Register List

- **INTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INTS1	INTS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **INTS1~INTS0**: Interrupt edge control for INT 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	–	TF	D5	INTF	TE	D2	INTE	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 **TF**: Timer/Event Counter interrupt request flag
0: No request
1: Interrupt request

Bit 5 **D5**: Reserved, cannot be changed

Bit 4 **INTF**: INT interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TE**: Timer/Event Counter interrupt control
0: Disable
1: Enable

Bit 2 **D2**: Reserved, cannot be changed

Bit 1 **INTE**: INT 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	—	RELDF	EMPTF	TBF	—	RELDE	EMPTE	TBE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **RELDF**: Reload data interrupt request flag
0: No request
1: Interrupt request

Bit 5 **EMPTF**: Buffer empty interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TBF**: Time Base interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **RELDE**: Reload data interrupt control
0: Disable
1: Enable

Bit 1 **EMPTE**: Buffer empty interrupt control
0: Disable
1: Enable

Bit 0 **TBE**: Time Base interrupt control
0: Disable
1: Enable

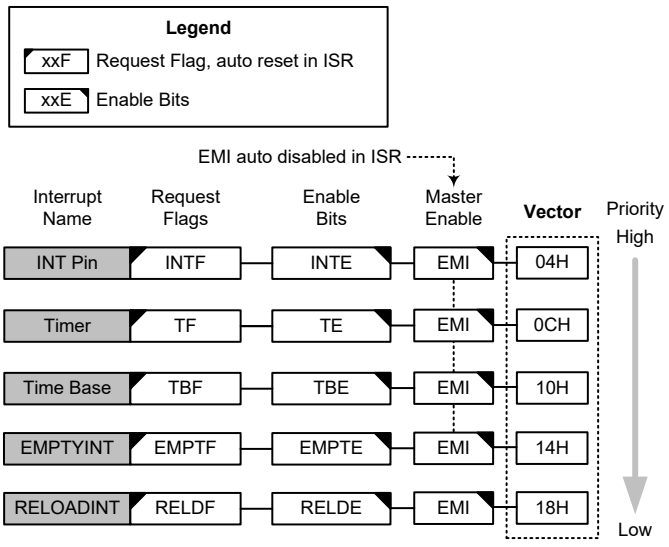
Interrupt Operation

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

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

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

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



Interrupt Structure

External Interrupt

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

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

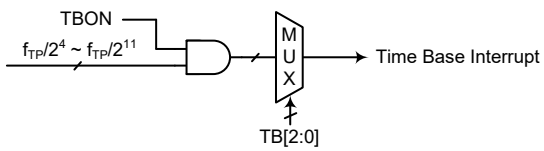
Timer/Event Counter Interrupt

An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TF, is set, which occurs when the Timer/Event Counter overflows. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Timer/Event Counter Interrupt enable bit, TE, must first be set. When the interrupt is enabled, the stack is not full and the Timer/Event Counter overflows, a subroutine call to its interrupt vector will take place. When the interrupt is serviced, the Timer/Event Counter Interrupt flag, TF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Time Base Interrupt

The function of the Time Base Interrupt is to provide regular time signals in the form of an internal interrupt. It is controlled by the overflow signals from the timer function. When this happens its interrupt request flag TBF will be set. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and Time Base enable bit, TBE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to its vector location will take place. When the interrupt is serviced, the interrupt request flag TBF 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_{TP}, originates from the internal clock source f_{SYS} or f_{SUB}, selected by setting the TS bit and then passes through a divider, which has been described in the Timer/Event Counter chapter. The divided clock source for the Timer Base function is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges.



Time Base Interrupt

• **TBC Register**

Bit	7	6	5	4	3	2	1	0
Name	TBON	—	—	—	—	TB2	TB1	TB0
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TBON**: Time Base control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB2~TB0**: Time Base time-out period selection
000: $2^4/ f_{TP}$
001: $2^5/ f_{TP}$
010: $2^6/ f_{TP}$
011: $2^7/ f_{TP}$
100: $2^8/ f_{TP}$
101: $2^9/ f_{TP}$
110: $2^{10}/ f_{TP}$
111: $2^{11}/ f_{TP}$

Buffer Empty Interrupt

A buffer empty interrupt request will take place when the buffer empty interrupt request flag, EMPTF, is set, which occurs when the buffer is empty. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the buffer empty interrupt enable bit, EMPTE, must first be set. When the interrupt is enabled, the stack is not full and an empty buffer situation occurs, a subroutine call to the buffer empty interrupt vector, will take place. When the interrupt is serviced, the buffer empty interrupt flag, EMPTF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Reload data Interrupt

A reload data interrupt request will take place when the reload data interrupt request flag, RELDF, is set, which occurs when the hardware has completed the PLADH/PLADL reload. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the reload data interrupt enable bit, RELDE, must first be set. When the interrupt is enabled, the stack is not full and a reload data interrupt request will take place when the reload data interrupt request flag, RELDF, is set, which occurs when the PLADH/PLADL reload situation occurs, a subroutine call to the reload data interrupt vector, will take place. When the interrupt is serviced, the reload data interrupt flag, RELDF, 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 devices are in the SLEEP or IDLE Mode and their system oscillator stopped, situations such as external edge transitions on the external interrupt pin may cause their 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 their enter 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 SLEEP or IDLE Mode, the wake-up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

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

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

Configuration Options

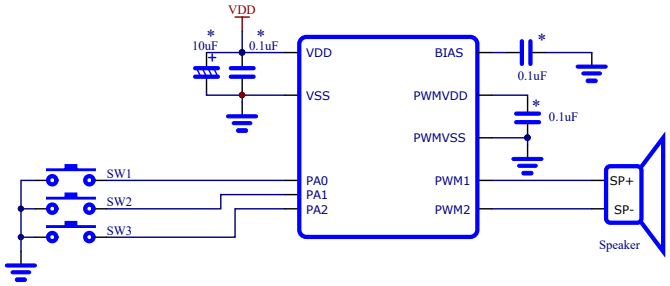
Configuration options refer to certain options within the MCU that are programmed into the devices during the programming process. During the development process, these options are selected using the HT-IDE software development tools. All options must be defined for proper system function, the details of which are shown in the table.

No.	Option
Oscillator Option	
1	HIRC frequency selection – f_{HIRC}: 8MHz, 12MHz or 16MHz

Note: 1. The HIRC frequency selected by the configuration option should be the same as the frequency determined by Holtek Voice MCU Workshop.

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

Application Circuits



Note: The capacitors marked with * should be placed close to the devices.

Peripheral IC Mode

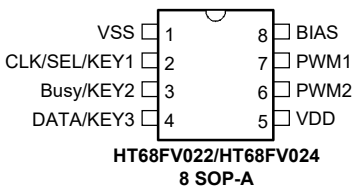
The devices provide the Peripheral mode for editing voice using Holtek Voice MCU Workshop. They also have three interface control modes, implementing both voice data and control interface program programming, as described below.

Features for Peripheral IC Mode

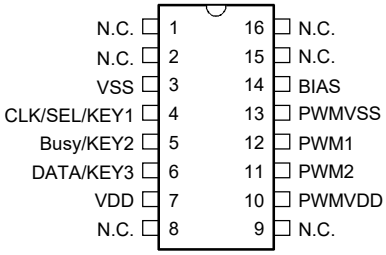
- Operating voltage (V_{DD}): 2.3V~5.5V
- Standby current about 1.5μA @ V_{DD}=3.3V, 2.5μA @ V_{DD}=5V
- Low Voltage Reset function – LVR
- Three control modes: One-wire, Two-wire and Direct mode
- PWM Voice output – can drive 8Ω/0.5W speaker directly

  Note: When setting the volume for direct drive speaker, the PWM output current and speaker wattage should be taken into consideration, to avoid damage to the speaker
- PWM Voice for both Normal Mode and Green Mode
- Voice Coding modes: ADPCM, u-Law, PCM
- Busy State output function
- Voice sampling rate supports up to 44kHz Input
- Voice and Sentence arrangement
- Maximum voice time: HT68FV022: 400s; HT68FV024: 800s
- Holtek Voice MCU Workshop tool to aid development
- Package types: 8-pin SOP, 16-pin NSOP

Pin Assignment for Peripheral IC Mode



HT68FV022/HT68FV024
8 SOP-A



HT68FV022/HT68FV024
16 NSOP-A

Pin Descriptions for Peripheral IC Mode

Pin Name	Function	Type	Description
CLK/SEL/KEY1 (PA0)	CLK	I	Two-wire interface CLK signal
	SEL	I	One-wire or Two-wire interface select
	KEY1	I	KEY1 input for Direct mode – active low
Busy/KEY2 (PA2)	Busy	O	Output low when voice playing – One-wire or Two-wire mode
	KEY2	I	KEY2 input for Direct mode – active low
DATA/KEY3 (PA1)	DATA	I	One-wire or Two-wire interface DATA signal
	KEY3	I	KEY3 input for Direct mode – active low
PWM1 (PB0)	PWM1	O	Voice PWM output 1
PWM2 (PB1)	PWM2	O	Voice PWM output 2
BIAS	BIAS	PWR	BIAS voltage, connect a 0.1μF capacitor to ground
VDD	VDD	PWR	Positive power supply
VSS	VSS	PWR	Negative power supply, ground
PWMVDD	PWMVDD	PWR	PWM positive power supply
PWMVSS	PWMVSS	PWR	PWM negative power supply, ground

Legend: I: Input; O: Output; PWR: Power

Control Modes

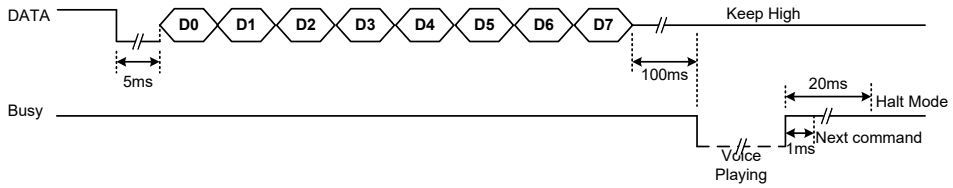
The devices have various control modes, which are the key controlled Direct mode and serial interface controlled One-wire and Two-wire modes. The control mode can be set by users according to their requirements when arranging the voices in Holtek Voice MCU Workshop. The Direct mode provides three keys.

In addition, the SEL pin can be used to select the One-wire mode or Two-wire mode by directly connecting an external resistor between the pin and ground.

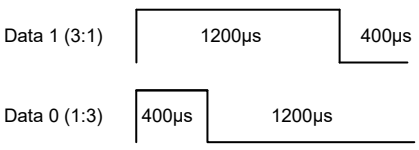
Serial Interface Control Modes

The devices support three serial interface control modes, the One-wire mode, Two-wire mode 1 and Two-wire mode 2. Their control timing and commands are described below.

One-wire Mode



After DATA is pulled low for a time range of 4ms~15ms (5ms recommended), 8 bits of data will be sent, LSB first and MSB last. The value of each bit is expressed in terms of the ratio of high voltage to low voltage.



When the high to low ratio is 3:1, this indicates a value of 1. It is recommended to use the values 1200μs:400μs.

When the high to low ratio is 1:3, this indicates a value of 0. It is recommended to use the values 400μs:1200μs.

After command end the DATA line needs to be kept at a high level.

Maximum 100ms from DATA end to Busy pulled low.

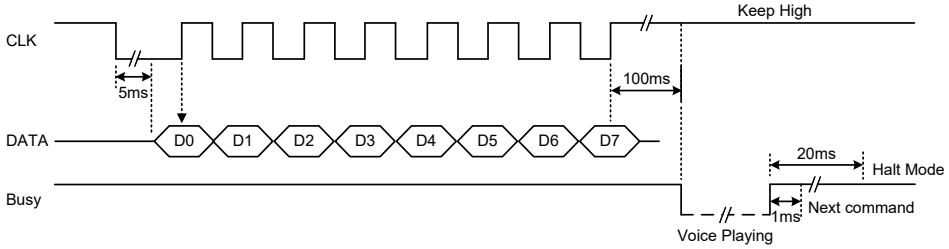
Maximum 20ms from Voice end into Halt mode.

At least 1ms from BUSY end to the next command.

Value range: 60μs:180μs ~ 600μs:1800μs. Note that it is recommended to use 3:1 and 1:3 voltage ratios to ensure communication stability.

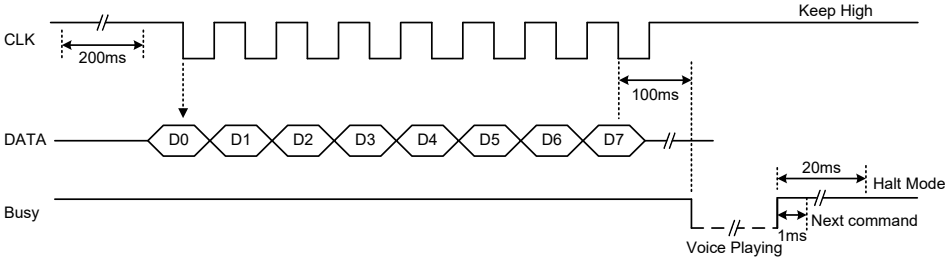
Two-wire Mode

- Two-wire Mode 1



After CLK is pulled low for 4ms~15ms (5ms is recommended), CLK will fetch the DATA on the rising edge and 8 bits of data will be sent, LSB first and MSB last.
The CLK period range is 200μs~5ms, a value of 600μs is recommended.
After command end the CLK line needs to be kept at a high level.
Maximum 100ms from CLK end to Busy pulled low.
Maximum 20ms from Voice end into Halt mode.
At least 1ms from BUSY end to the next command.

- Two-wire Mode 2



After CLK is pulled high for 200ms~400ms (200ms is recommended), CLK will fetch the DATA on the falling edge and 8 bits of data will be sent, LSB first and MSB last.
The CLK period range is 8ms~30ms, a value of 20ms is recommended.
After command end the CLK line needs to be kept at a high level.
Maximum 100ms from CLK end to Busy pulled low.
Maximum 20ms from Voice end into Halt mode.
At least 1ms from BUSY end to the next command.

Control Commands

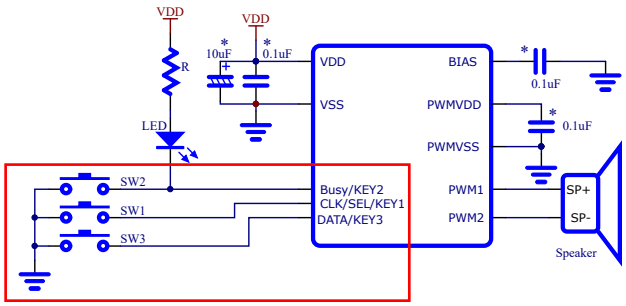
Command Code (Hexadecimal)	Function	Description
00H~7FH	Play voice	When the FAH command is used, select the desired voice to play, 00H is voice 0 01H is voice 1 And so on, from 0 to 127, there are 128 voices When the FBH command is used, select the desired voice to play, 00H is voice 128 01H is voice 129 And so on, from 128 to 255, there are 128 voices.
80H~DFH	Play sentence	Select the desired sentence to play, 80H is sentence 0 81H is sentence 1 And so on, from 0 to 95, there are 96 sentences.
E0H	Operating voltage selection	This command is used to control the LDO for supplying Voice Flash Memory power. When the operating voltage is 2.3V~3.6V, this command is required to adjust the Voice Flash Memory power voltage. If this command is not used, the default operating voltage will be 3.6V~5.5V. When applied to situations where the operating voltage varies, this command can be used to make the system operate under optimum conditions.
E1H~EFH	Volume selection	E1 is the minimum volume and EF is the maximum volume. There are 12 levels of volume adjustment. Note: The relationship between the voice command code and the PWDC[3:0] value is shown in the table below.
F1H	Pause voice/sentence	Pause playing the current voice and sentence.
F2H	Play after pause	Continue playing the paused voice and sentence.
F4H	Loop playback the current voice/sentence	Loop playback for the current voice and sentence.
F8H	Stop playing the current voice/sentence	Stop playing the current voice and sentence.
FAH (Voice extended command)	00H~7FH command is used to play voice	When the FAH command is used, the 00H~7FH command can play voice from 0 to 127
FBH (Voice extended command)		When the FBH command is used, the 00H~7FH command can play voice from 128 to 255

Direct Mode

The devices support a key control mode, namely the Direct mode, which can be set according to requirements when arranging voices in the Holtek Voice MCU Workshop. This mode supports three keys, KEY1~KEY3, which are active low. The key functions are as follows:

Key Pin	Function	Description
KEY1	Stop/Reset	Short press the key to stop playing the current voice, long press (3 seconds) the key to reset to the first voice.
KEY2	Play/Next	Short press the key to play voice, press again to play the next voice without requiring the current voice to finish. Long press (3 seconds) the key to play sentence, long press again to play the next sentence without requiring the current voice to finish.
KEY3	Volume Up/Down	Short press the key to increment volume, long press (3 seconds) to decrement. If long press for more than 3 seconds, it will decrement the volume even more. Note: There are 12 levels of volume. The relationship between the volume level and the PWDC[3:0] value is shown in the table below.

The key connection is shown in the red box below.



The direct mode volume levels and control mode command code corresponding to the PWDC[3:0] values are shown in the following table.

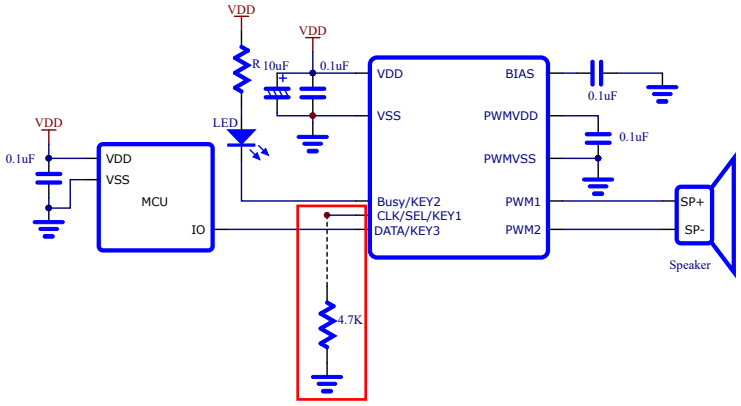
Direct Mode Volume Level	Control Mode Command Code	PWDC[3:0]
1	E1H	Mute (Ignore the PWDC[3:0] value)
2	E2H	0000B
3	E3H	0001B
4	E4H	0010B
5	E5H	0011B
6	E6H	0100B
7	E7H	0101B
8	E8H	0110B
9	E9H	0111B
10	EAH	1000B
11	EBH	1001B
12	ECH	1011B
13	EDH (for 16-pin NSOP only)	1100B
14	EEH (for 16-pin NSOP only)	1110B
15	EFH (for 16-pin NSOP only)	1111B

External Serial Interface Control Mode

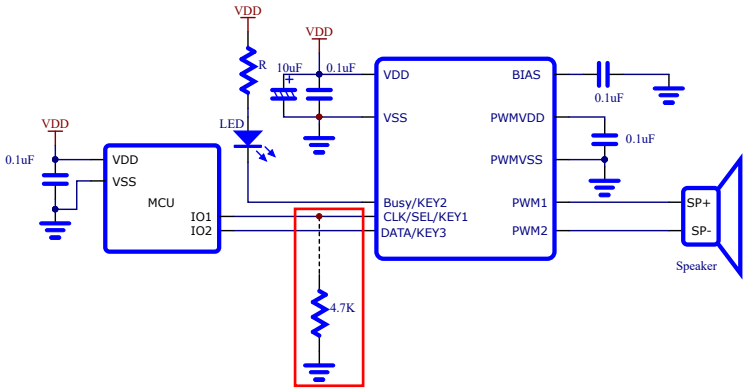
The control mode can be selected using the SEL pin. By connecting a 4.7kΩ resistor between the pin and ground, there are two ways to do this:

1. If the One-wire interface is selected in the Holtek Voice MCU Workshop, then it will change to the Two-wire mode 1 interface by connecting a 4.7kΩ resistor between the SEL pin and ground.
2. If the Two-wire interface is selected in the Holtek Voice MCU Workshop, then it will change to the One-wire mode interface by connecting a 4.7kΩ resistor between the SEL pin and ground.

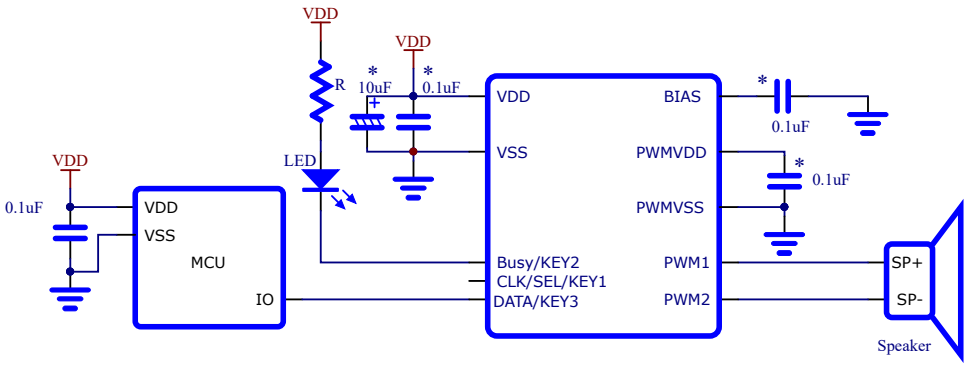
The original Two-wire mode will change to the One-wire mode after connecting a 4.7kΩ resistor to the SEL pin.



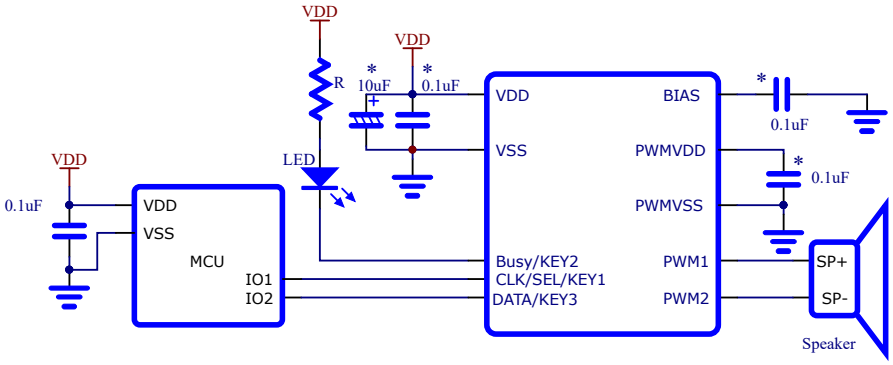
The original One-wire mode will change to the Two-wire mode 1 after connecting a 4.7kΩ resistor to the SEL pin.



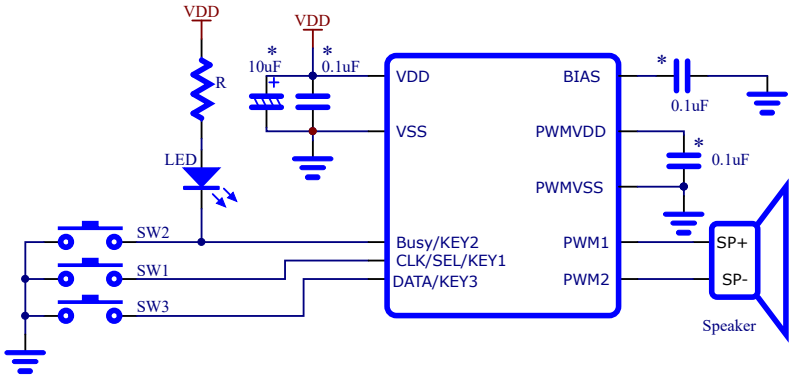
Application Circuits



One-wire Mode Application Circuit



Two-wire Mode Application Circuit



Direct Mode Application Circuit

Note: The capacitors marked with * should be placed close to the devices.

Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV, 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,[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].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 or current page) to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
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.

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

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

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

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

Affected flag(s) C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s): C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: ACC ← ACC − [m] − $\overline{C}$

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

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: [m] ← ACC − [m] − $\overline{C}$

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

SDZ [m] Skip if decrement Data Memory is 0

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

Operation:
[m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

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

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

Operation:
ACC ← [m] − 1
Skip if ACC=0

Affected flag(s): None

SET [m] Set Data Memory

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

Operation: [m] ← FFH

Affected flag(s): None

SET [m].i Set bit of Data Memory

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

Operation: [m].i ← 1

Affected flag(s): None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C, 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 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 or current page) to TBLH and Data Memory

Description The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

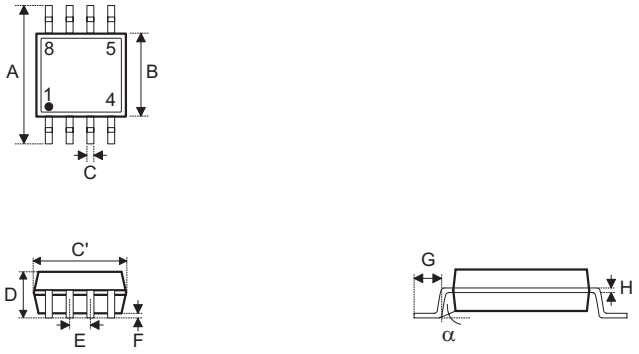
Package Information

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

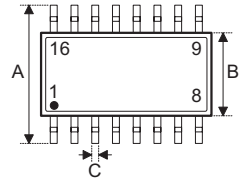
8-pin SOP (150mil) Outline Dimensions

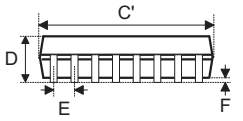


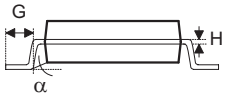
Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.012	—	0.020
C′		0.193 BSC	
D	—	—	0.069
E		0.050 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		6.00 BSC	
B		3.90 BSC	
C	0.31	—	0.51
C′		4.90 BSC	
D	—	—	1.75
E		1.27 BSC	
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

16-pin NSOP (150mil) Outline Dimensions







Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.012	—	0.020
C’		0.390 BSC	
D	—	—	0.069
E		0.050 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	0.31	—	0.51
C’		9.90 BSC	
D	—	—	1.75
E		1.27 BSC	
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°