HOLTEK

Dual Slope 16K/8K/4K ASSP MCU for Body Fat Scale

HT45R2K-C
HT45R2K-B
HT45R2K-A

Revision: 1.40 Date: March 12, 2015

Features

- Operating Voltage:
  - f_{SYS}= 4MHz: 2.2V~5.5V
  - f_{SYS}= 8MHz: 3.3V~5.5V
- Oscillator types:
  - External Crystal -- HXT
  - External Low Speed Crystal -- LXT
  - External RC -- ERC
  - Internal High Speed RC -- HIRC
  - Internal Low Speed RC -- LIRC
- Up to 22 bidirectional I/O lines
- One external interrupt input shared with an I/O lines
- One 8-bit and two 16-bit programmable timer/event counters with overflow interrupt a 8-stage prescaler
- LCD driver with 24×8 segments
- 16K×16 program memory
- 256×8 data memory
- Single differential input channel dual slope Analog to Digital Convertor with Operational Amplifier
- Watchdog Timer with regulator power
- Buzzer output
- HALT and wake-up functions to reduce power consumption
- Internal voltage regulator (3.3V) and charge pump
- Internal reference voltage generator (1.5V)
- 8-level subroutine nesting
- Bit manipulation instruction
- 16-bit table read instruction
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}= 5V
- 63 powerful instructions
- All instructions in 1 or 2 machine cycles
- Low voltage reset function
- One vibration sensor input
- Four touch-key inputs
- Body Fat circuit
- 80-pin LQFP package

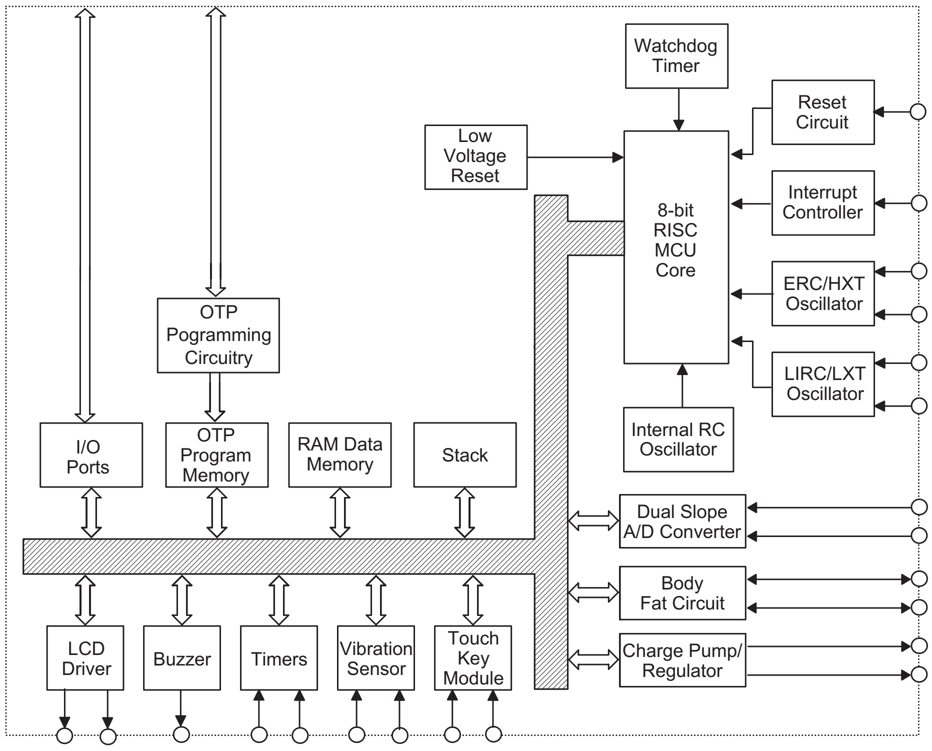
General Description

The HT45R2K-C is an 8-bit high performance, RISC architecture microcontroller device, which with its A/D Converter and LCD driver, can directly interface to analog signals and to LCD panels. The device includes a range of other features such as low power consumption, I/O flexibility, timer functions, oscillator options, dual slope A/D convertor, HALT and wake-up functions, watchdog timer, vibration sensor etc. However as the device includes all the circuitry associated with body fat measurement the device is especially suitable for this specific application area, being able to significantly reduce the need for the usual external components.

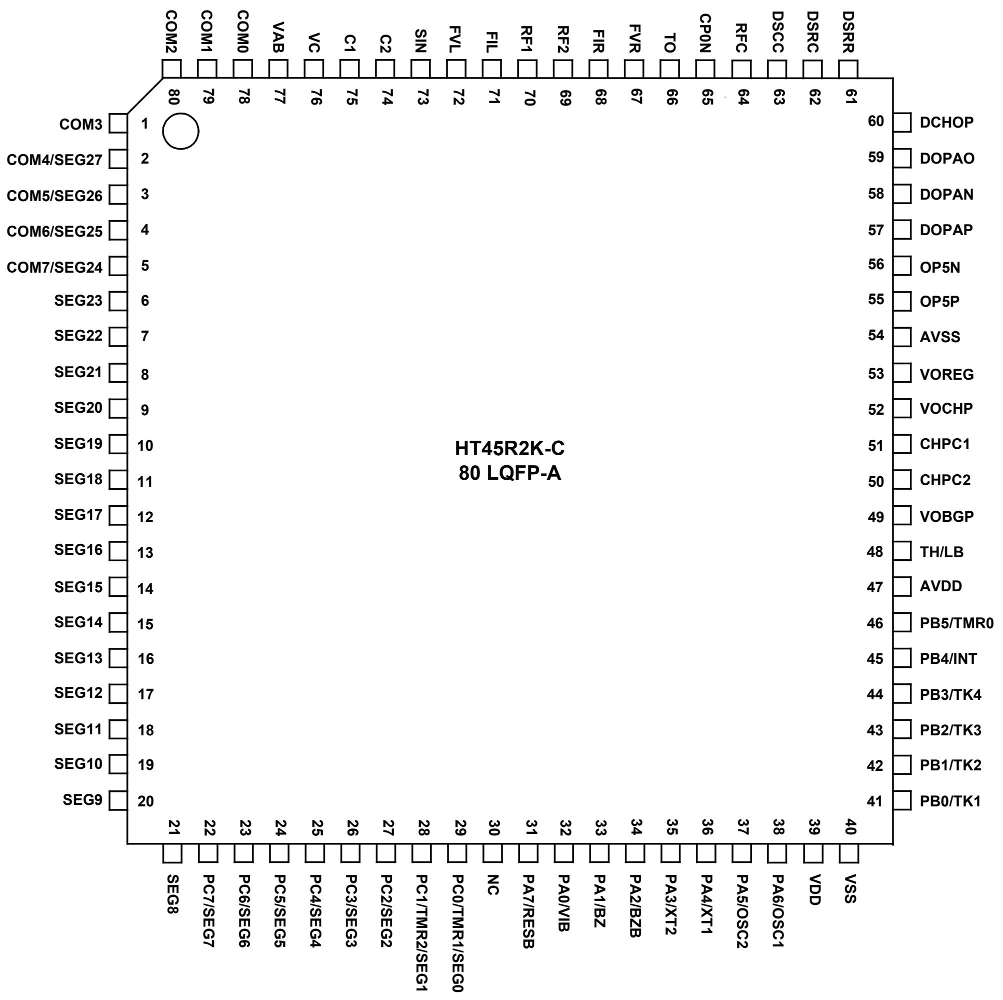
Selection Table

Part No.	Program Memory
HT45R2K-A	4K×16
HT45R2K-B	8K×16
HT45R2K-C	16K×16

Block Diagram



Pin Assignment



Pin Description

Pin Name	Function	OPT	I/T	O/T	Description
PA0/VIB	PA0	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	VIB	VIBRC	—	—	Vibration input
PA1/BZ	PA1	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	BZ	SFS	—	CMOS	Buzzer output
PA2/BZB	PA2	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	BZB	SFS	—	CMOS	Complementary buzzer output
PA3/XT2	PA3	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	XT2	CO	—	LXT	LXT pin
PA4/XT1	PA4	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	XT1	CO	LXT	—	LXT pin

Pin Name	Function	OPT	I/T	O/T	Description
PA5/OSC2	PA5	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OSC2	CO	—	HXT	HXT pin
PA6/OSC1	PA6	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OSC1	CO	HXT	—	HXT/ERC pin
PA7/$\overline{\text{RES}}$	PA7	PAWK	ST	NMOS	General purpose I/O. Register enabled wake-up.
	$\overline{\text{RES}}$	CO	ST	—	Reset input
PB0/TK1	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TK1	TKM0C1	—	—	Touch key 1 input
PB1/TK2	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TK2	TKM0C1	—	—	Touch key 2 input
PB2/TK3	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TK3	TKM0C1	—	—	Touch key 3 input
PB3/TK4	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TK4	TKM0C1	—	—	Touch key 4 input
PB4/INT	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	INT	CTRL1 INTC0	ST	—	External interrupt input
PB5/TMR0	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TMR0	TMR0C	ST	—	External Timer 0 clock input
PC0/TMR1/ SEG0	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TMR1	TMR1C	ST	—	External Timer 1 clock input
	SEG0	LCDOUT	—	CMOS	LCD segment output
PC1/TMR2/ SEG1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TMR2	TMR2C	ST	—	External Timer 2 clock input
	SEG1	LCDOUT	—	CMOS	LCD segment output
PC2/SEG2	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG2	LCDOUT	—	CMOS	LCD segment output
PC3/SEG3	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG3	LCDOUT	—	CMOS	LCD segment output
PC4/SEG4	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG4	LCDOUT	—	CMOS	LCD segment output
PC5/SEG5	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG5	LCDOUT	—	CMOS	LCD segment output
PC6/SEG6	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG6	LCDOUT	—	CMOS	LCD segment output
PC7/SEG7	PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SEG7	LCDOUT	—	CMOS	LCD segment output
SEG8~SEG23	SEGn	—	—	CMOS	LCD segment outputs
COM0~COM3	COMn	—	—	CMOS	LCD common outputs
COM4/SEG27	COM4	LCDC	—	CMOS	LCD common output
	SEG27				LCD segment output

Pin Name	Function	OPT	I/T	O/T	Description
COM5/SEG26	COM5	LCDC	—	CMOS	LCD common output
	SEG26				LCD segment output
COM6/SEG25	COM6	LCDC	—	CMOS	LCD common output
	SEG25				LCD segment output
COM7/SEG24	COM7	LCDC	—	CMOS	LCD common output
	SEG24				LCD segment output
VAB	VAB	—	—	NS	LCD voltage pump
VC	VC	—	—	NS	LCD voltage pump
C1	C1	—	—	NS	LCD voltage pump
C2	C2	—	—	NS	LCD voltage pump
VOBGP	VOBGP	—	—	—	Band gap voltage output pin (for internal use)
VOREG	VOREG	—	—	PWR	Regulator output 3.3V
VOCHP	VOCHP	—	—	PWR	Charge pump output
CHPC1	CHPC1	—	—	NS	Charge pump capacitor – Positive
CHPC2	CHPC2	—	—	NS	Charge pump capacitor – Negative
DOPAN	DOPAN	—	NS	—	OPA Negative input
DOPAP	DOPAP	—	NS	—	OPA Positive input
DOPAO	DOPAO	—	—	NS	OPA output
DCHOP	DCHOP	—	—	NS	OPA Chopper pin
TH/LB	TH	—	NS	—	Temperature sensor input
	LB	—	NS	—	Low battery voltage input
DSRR	DSRR	—	NS	—	Reference signal input
DSRC	DSRC	—	NS	—	Integrator negative input
DSCC	DSCC	—	NS	—	Comparator negative input
SIN	SIN	—	—	NS	Sine wave output
FVR	FVR	—	NS	—	Foot resistor channel
FVL	FVL	—	NS	—	Foot resistor channel
FIL	FIL	NS	—	—	Foot resistor channel
FIR	FIR	—	NS	—	Foot resistor channel
RF1	RF1	—	—	NS	Reference impedance channel
RF2	RF2	—	—	NS	Reference impedance channel
RFC	RFC	—	—	—	ADC analog input
T0	T0	—	—	—	OPAC output
CP0N	CP0N	—	NS	—	The inverting input of CP0
OP5N	OP5N	—	NS	—	The inverting input of OP5
OP5P	OP5P	—	NS	—	The non-inverting input of OP5
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground
AVDD	AVDD	—	PWR	—	Analog power supply
AVSS	AVSS	—	PWR	—	Analog ground

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

Absolute Maximum Ratings

Supply Voltage	V_{SS} -0.3V to +6.0V
Input Voltage	V_{SS} -0.3V to V_{DD} +0.3V
Storage Temperature	-50°C to +125°C
Operating Temperature	-40°C to +85°C
I_{OL} Total	150mA
I_{OH} Total	-100mA
Total Power Dissipation	500mW

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

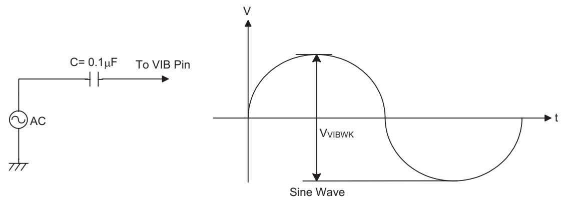
D.C. Characteristics

Operating Temperature: -40°C °C to 85°C Ta= 25°C Typical

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		VDD	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}= 4MHz	2.2	—	5.5	V
		—	f_{SYS}= 8MHz	3.3	—	5.5	V
V_{LCD}	LCD Highest Voltage	—	—	0	—	V_{DD}	V
I_{DD1}	Operating Current (HXT, ERC,HIRC)	5V	No load, f_{SYS}= 8MHz ADC off	—	4	8	mA
I_{DD2}	Operating Current (HXT, ERC,HIRC)	3V	No load, f_{SYS}= 4MHz ADC off	—	0.8	1.5	mA
		5V		—	2.5	4.0	mA
I_{DD3}	Operating Current (HXT, ERC)	3V	No load, f_{SYS}= 2MHz ADC off	—	0.5	1.0	mA
		5V		—	1.5	3.0	mA
I_{DD4}	Operating Current (HXT, ERC,HIRC)	5V	V_{OREG}= 3.3V, f_{SYS}= 4MHz ADC on, ADC clock is 125kHz (all other analog devices off)	—	3	5	mA
I_{STB1}	Standby Current	3V	No load, system HALT, ADC, LCD and WDT off	—	—	1	μA
		5V		—	—	2	μA
I_{STB2}	Standby Current	3V	No load, system HALT, ADC and LCD off, WDT on	—	2.5	5.0	μA
		5V		—	8	15	μA
I_{STB3}	Standby Current (Internal RC 12kHz)	3V	No load, system osc off, ADC and WDT off	—	2	5	μA
		5V		—	6	10	μA
I_{STB4}	Standby Current (Internal RC 12kHz)	5V	No load, system osc off, ADC and WDT off LCD ON(1/3 R type)	—	380	500	μA
I_{STB5}	Standby Current	3V	No load, system HALT, LXT osc slowly start-up, WDTOSC off, LXT on	—	—	5	μA
		5V		—	—	15	μA
I_{STB6}	Standby Current (LXT)	5V	No load, system osc off, ADC and WDT off, LCD on (1/3 R type), (LCD Bias current= 50μA)	—	390	510	μA
I_{STB7}	Standby Current	3V	No load, Only vibration sensor turn on&VIB pin connected a 0.1μF cap to VSS, WDT off	—	2	4	μA
		5V		—	8	16	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		VDD	Conditions				
V_{IL1}	Input Low Voltage for I/O Ports, TMR0, TMR1, TMR2 and INT	5V	—	0.0	—	1.5	V
		—	—	0.0	—	$0.2V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR0, TMR1, TMR2 and INT	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	$1.0V_{DD}$	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
V_{LVR1}	Low Voltage Reset	—	Configuration Option: 2.1V	2.0	2.1	2.2	V
V_{LVR2}			Configuration Option: 2.55V	2.40	2.25	2.70	V
V_{LVR3}			Configuration Option: 3.15V	3.00	3.15	3.30	V
V_{LVR4}			Configuration Option: 3.8V	3.6	3.8	4.0	V
I_{OL1}	I/O Port Sink Current	3V	V_{OL}= $0.1V_{DD}$	4	8	—	mA
		5V		10	20	—	mA
I_{OH1}	I/O Port Source Current	3V	V_{OH}= $0.9V_{DD}$	-2	-4	—	mA
		5V		-5	-10	—	mA
I_{OL2}	LCD Common and Segment Sink Current	3V	V_{OL}= $0.1V_{DD}$	210	420	—	μA
		5V		350	700	—	μA
I_{OH2}	LCD Common and Segment Source Current	3V	V_{OH}= $0.9V_{DD}$	-80	-160	—	μA
		5V		-180	-360	—	μA
I_{OL3}	PA7 Sink Current	5	V_{OL}= $0.1V_{DD}$	2	3	—	mA
R_{PH}	Pull-high Resistance of I/O Ports	3V	—	20	60	100	KΩ
		5V	—	10	30	50	KΩ
V_{VIBWK}	Minimum Voltage to Wake MCU by the Vibration Sensor Input	2.4V	100Hz~1kHz sine wave (Note)	250	—	—	mV
		3V					
		5V					
Charge Pump and Regulator							
V_{CHP}	Input Voltage	—	Charge Pump on	2.2	—	3.6	V
		—	Charge Pump off	3.7	—	5.5	V
V_{OREG}	Output Voltage	—	No load	3.0	3.3	3.6	V
V_{REGDP1}	Regulator Output Voltage Drop (Compared with No Load)	—	V_{DD} = 3.7V~5.5V Charge Pump off Current<= 10mA	—	100	—	mV
V_{REGDP2}		—	V_{DD} = 2.4V~3.6V Charge Pump on Current <= 6mA	—	100	—	mV
Dual Slope AD Convertor, Amplifier and Band Gap							
V_{ADOFF}	Input Offset Range	—	V_{OREG}= 3.3V	—	500	800	μV
V_{RFGTC}	Reference Generator Temperature Coefficient	—	V_{OREG}= 3.3V	—	50	—	ppm/C
V_{ICMR}	Common Mode Input Range	—	Amplifier, No load	0.2	—	V_{OREG}-1.2	V
		—	Integrator, No load	1.2	—	V_{OREG}-0.2	V

Note: Test Circuit for V_{VMBWK}.



A.C. Characteristics

Operating Temperature: -40°C to 85°C Ta= 25°C Typical

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		VDD	Conditions				
f_{SYS}	System Clock (RC)	2.2V~5.5V	—	400	—	4000	kHz
	System Clock (HXT)	2.2V~5.5V	—	400	—	4000	kHz
		3.3V~5.5V	—	400	—	8000	kHz
		4.5V~5.5V	—	400	—	12000	kHz
f_{HIRC}	HIRC OSC	3V/5V	Ta= 25°C	-2%	4	+2%	MHz
		3V/5V	Ta= 25°C	-2%	8	+2%	MHz
		5V	Ta= 25°C	-2%	12	+2%	MHz
		3V/5V	Ta= 0~70°C	-5%	4	+5%	MHz
		3V/5V	Ta= 0~70°C	-5%	8	+5%	MHz
		5V	Ta= 0~70°C	-5%	12	+5%	MHz
		2.2V~3.6V	Ta= 0~70°C	-8%	4	+8%	MHz
		3.0V~5.5V	Ta= 0~70°C	-8%	4	+8%	MHz
		3.0V~5.5V	Ta= 0~70°C	-8%	8	+8%	MHz
		4.5V~5.5V	Ta= 0~70°C	-8%	12	+8%	MHz
		2.2V~3.6V	Ta= -40~85°C	-12%	4	+12%	MHz
		3.0V~5.5V	Ta= -40~85°C	-12%	4	+12%	MHz
		3.0V~5.5V	Ta= -40~85°C	-12%	8	+12%	MHz
		4.5V~5.5V	Ta= -40~85°C	-12%	12	+12%	MHz
f_{ERC}	ERC OSC	5V	Ta= 25°C, R= 120K	-2%	4	+2%	MHz
		5V	Ta= 0~70°C, R= 120K	-5%	4	+5%	MHz
		5V	Ta= -40~85°C, R= 120K	-7%	4	+7%	MHz
		2.2V~5.5V	Ta= -40~85°C, R= 120K	-11%	4	+11%	MHz
f_{TIMER}	Timer I/P Frequency (TMR0,TMR1,TMR2)	2.2V~5.5V	—	0	—	4000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period (Wake-up from HALT)	—	—	2	1024	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1.00	2.00	ms

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

2. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS and located as close to the device as possible.

Operational Amplifier Electrical Characteristics

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
DC Electrical Characteristics							
V_{DD}	Supply Voltage	—	—	2.2	—	5.5	V
I_{CC}	Supply Current Per Signal Amplifier	5V	No load	150	360	500	μA
OP0, OP2							
SR	Slew Rate at Unity Gain	3V	R_L= 100KΩ, C_L= 100pF	7.5	—	—	V/μs
GBW	Gain Bandwidth Product	3V	R_L= 100KΩ, C_L= 100pF	—	—	2	MHz
OP1							
SR	Slew Rate at Unity Gain	3V	R_L= 100KΩ, C_L= 100pF	7.5	—	—	V/μs
GBW	Gain Bandwidth Product	3V	R_L= 100KΩ, C_L= 100pF	—	—	5	MHz

Instrumentation Amplifier Electrical Characteristics

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
V_{DD}	Supply Voltage	—	—	2.2	—	5.5	V
Icc	Supply Current	5V	Io= 0A	—	1.2	—	mA
SR	Slew Rate at Unity Gain	3V	RL= 100KΩ, CL= 100pF	7.5	—	—	V/μs
GBW	Gain Bandwidth Product	3V	RL= 100 KΩ, CL= 100pF	—	—	5	MHz
GER	Gain Error	—	—	-10%	—	10%	
VOS	Input Offset Voltage	3V	—	-15	—	15	mV

Power-on Reset Characteristics

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{POR}	VDD Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR_{VDD}	VDD Rising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
V_{DCPOR_MAX}	DC POR Maximum Voltage	—	—	0.6	—	2.2	V
t_{POR1}	Minimum Time for VDD Stays at VPOR to Ensure Power-on Reset	—	Without 0.1μF between VDD and VSS	2	—	—	μs
t_{POR2}	Minimum Time for VDD Stays at VPOR to Ensure Power-on Reset	—	With 0.1μF between VDD and VSS	10	—	—	μs

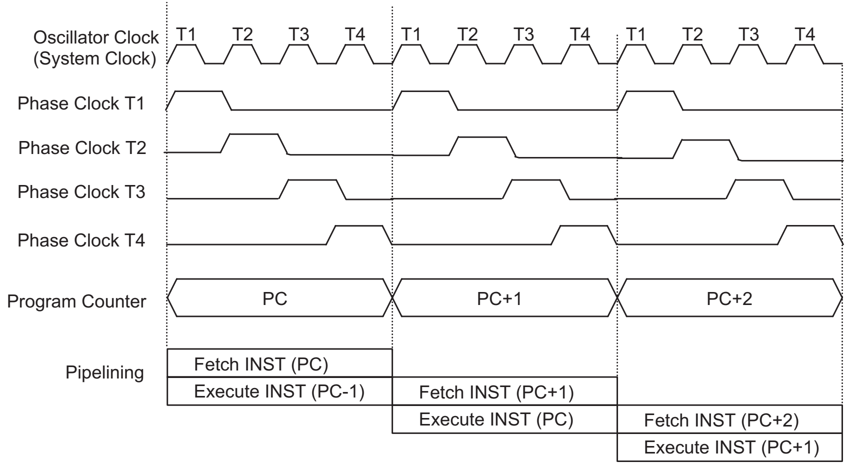
System Architecture

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

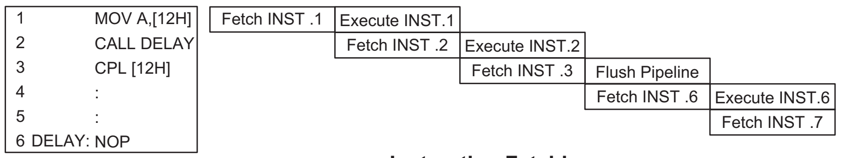
Clocking and Pipelining

The system clock, derived from an RC 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 instruction 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



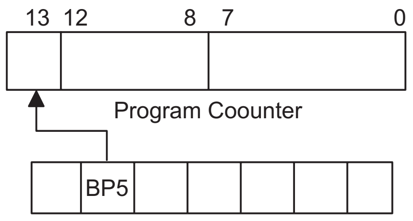
Instruction Fetching

Program Counter – PC

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

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

Program Counter	
Program Counter High Byte	PCL Register Low Byte
PC13~PC8	PCL7~PCL0



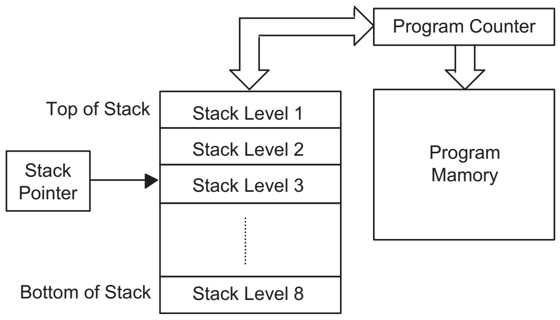
Bank Pointer(BP)

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

The lower byte of the Program Counter is fully accessible under program control. Manipulating the PCL might cause program branching, so an extra cycle is needed to pre-fetch. Further information on the PCL register can be found in the Special Function Register section.

Stack

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



If the stack is full and an enabled interrupt takes place, the interrupt request flag will be recorded but 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.

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

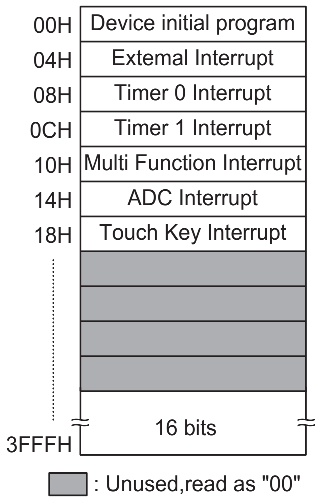
Program Memory

The Program Memory is the location where the user code or program is stored. The device is supplied with One-Time Programmable, OTP, memory where users can program their application code into the device. By using the appropriate programming tools, OTP devices offer users the flexibility to freely develop their applications which may be useful during debug or for products requiring frequent upgrades or program changes.

Structure

The Program Memory has a capacity of 16K×16. 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 separate table pointer registers.

The device has its Program Memory divided into two Banks, Bank 0 and Bank 1. The required Bank is selected using Bit 5 of the BP Register.



Program Memory Structure

Special Vectors

Within the Program Memory, certain locations are reserved for special usage such as reset and interrupts.

- **Reset Vector**

  This vector is reserved for use by the device reset for program initialization. After a device reset is initiated, the program will jump to this location and begin execution.

- **External interrupt vector**

  This vector is used by the external interrupt. If the external interrupt pin on the device receives an edge transition, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full. The external interrupt active edge transition type, whether high to low, low to high or both is specified in the CTRL1 register.

- **Timer/Event 0/1 counter interrupt vector**

  This internal vector is used by the Timer/Event 0/1 Counters. If a Timer/Event Counter overflow occurs, the program will jump to its respective location and begin execution if the associated Timer/Event Counter interrupt is enabled and the stack is not full.

- **Multi-function interrupt vector**

  This vector is used by the Multi-function interrupt. If the Timer/Event 2 Counter overflow or Touch Key 16-bit or 10-bit Counter overflow, the program will jump to this location and begin execution if the relevant interrupt is enabled and the stack is not full.

- **A/D Convertor interrupt vector**

  This internal vector is used by the A/D Convertor interrupt. If the A/D conversion process finishes, the program will jump to this location and begin execution if the A/D Convertor interrupt is enabled and the stack is not full.

- **Touch Key interrupt vector**

  This internal vector is used by the Touch Key interrupt. If the counter in the relevant Touch Key module overflow occurs, the program will jump to this location and begin execution if the Touch Key interrupt is enabled and the stack is not full.

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

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

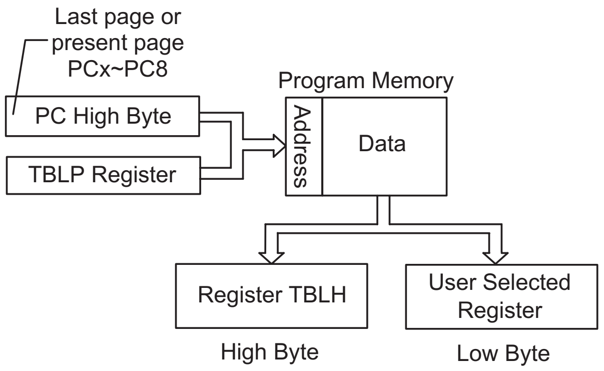


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "3F00H" which refers to the start address of the last page within the 16K words Program Memory of the device. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "3F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRDC [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 "TABRDL [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.

Instruction(s)	Table Location													
	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	PC13	PC12	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: 1. PC13~PC8: Current Program Counter bits
2. @7~@0: Table Pointer TBLP bits
3. b13~b0: Table address location bits

Table Read Program Example

```
tempreg1 db ?              ;temporary register #1
tempreg2 db ?              ;temporary register #2
:
:
mov a,06h                  ;initialise low table pointer - note that this
                           ;address
mov tblp,a                 ;is referenced
:
:
tabrdl tempreg1            ;transfers value in table referenced by table
                           ;pointer data at program
                           ;memory address "3F06H" transferred to tempreg1
                           ;and TBLH
dec tblp                   ;reduce value of table pointer by one tabrdl
                           ;tempreg2
                           ;transfers value in table referenced by table
                           ;pointer data at program
                           ;memory address "3F05H" transferred to tempreg2
                           ;and TBLH in this
                           ;example the data "1AH" is transferred to tempreg1
                           ;and data "0FH" to register tempreg2
:
:
org 3F00h                  ;sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
```

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

The data memory is divided into four banks, Bank0~3. The Bank0 is subdivided into two sections, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.

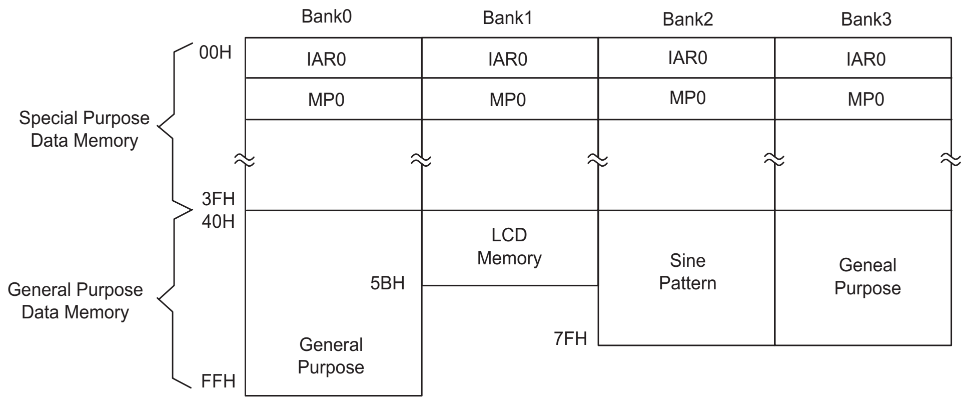
The general purpose data memory Bank1 and Bank2 are dedicated for the LCD and Sine Pattern functions respectively while the Bank3 is used for general purpose data memory. The total capacity of general purpose data memory, available for designer, is 256 bytes, composed by Bank0 and Bank4. The accompanying table illustrates the Data Memory arrangement.

Capacity	Bank	Description
256×8	Bank 0(00H~3FH)	Special Function Data Memory(64Bytes)
	Bank 0(40H~FFH)	General Purpose Data Memory(192 Bytes)
	Bank 1(40H~5BH)	General Purpose Data Memory for LCD(28 Bytes)
	Bank 2(40H~7FH)	General Purpose Data Memory for Sine Pattern(64 Bytes)
	Bank 3(40H~7FH)	General Purpose Data Memory(64 Bytes)

Note: The Data RAM Capacity consists of the general purpose data RAM in Bank0 and Bank3, except the Bank1 and Bank2 which are assigned to the LCD and Sine Pattern functions.

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 program for both read and write operations. By using the "SET [m].i" and "CLR [m].i" 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.

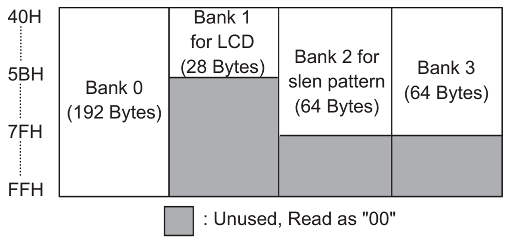
For this device, the Data Memory is divided into four banks, which are selected using a Bank Pointer. Only data in Bank 0 can be directly addressed, data in Bank 1~3 must be indirectly addressed.



Data Memory Structure

	Bank 0, Bank 1, Bank 2, Bank 3
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	CTRL0
0AH	STATUS
0BH	INTC0
0CH	LVRC
0DH	TMR0
0EH	TMR0C
0FH	TMR1H
10H	TMR1L
11H	TMR1C
12H	PA
13H	PAC
14H	PB
15H	PBC
16H	DACO
17H	FTRC
18H	ADCR
19H	SWC
1AH	ADCD
1BH	WDTC1
1CH	WDTC
1DH	WDTD
1EH	INTC1
1FH	CHPRC
20H	TMR2H
21H	TMR2L
22H	TMR2C
23H	SGC
24H	SGDNR
25H	SGN
26H	HALTC
27H	LCDOUT
28H	CTRL1
29H	VIBRC
2AH	IADAC
2BH	PC
2CH	PCC
2DH	PAWK
2EH	PAPU
2FH	PBPU
30H	PCPU
31H	SFS
32H	OPAC
33H	LCDC
34H	MFIC
35H	IAC0
36H	IAC1
37H	TKM0C4
38H	TKM016DH
39H	TKM016DL
3AH	TKM010DL
3BH	TKM0RO
3DH	TKM0C0
3CH	TKM0C1
3EH	TKM0C2
3FH	TKM0C3

Special Purpose Data Memory



General Purpose 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 Function Registers

To ensure successful operation of the microcontroller, certain internal registers are implemented in the Data Memory area. These registers ensure correct operation of internal functions such as timers, interrupts, etc., as well as external functions such as I/O data control. The locations of these registers within the Data Memory begin at the address “00H” and are mapped into from Bank 0 to Bank 3. Any unused Data Memory locations between these special function registers and the point where the General Purpose Memory begins is reserved and attempting to read data from these locations will return a value of “00H”.

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 and MP0 can together access data from Bank 0 while the IAR1 and MP1 register pair can access data from any bank. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of “00H” and writing to the registers indirectly will result in no operation.

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to indirectly address and track data. MP0 can only be used to indirectly address data in Bank 0 while MP1 can be used to address data from Bank 0 to Bank 3. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, is the address specified by the related Memory Pointer. Note that indirect addressing using MP1 and IAR1 must be used to access any data in Bank 1~Bank 3.

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

Indirect Addressing Program Example

```
data .section ´data´
adres1      db ?
adres2      db ?
adres3      db ?
adres4      db ?
block       db ?
code .section at 0 ´code´
org00h
start:
    mov a, 04h             ;setup size of block
    mov block, a
    mov a, offset adres1   ;Accumulator loaded with first RAM address
    mov mp0, a             ;setup memory pointer with first RAM address
loop:
    clr IAR0               ;clear the data at address defined by mp0
    inc mp0                ;increment memory pointer
    sdz block              ;check if last memory location has been cleared
    jmp loop
continue:
```

The important point to note here is that in the example, 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 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.

Bank Pointer – BP

In this device, the Program and Data Memory are divided into several banks. Selecting the required Program and Data Memory area is achieved using the Bank Pointer. Bit 5 of the Bank Pointer is used to select Program Memory Bank 0 or 1, while bits 0~1 are used to select Data Memory Banks 0~3.

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

As both the Program Memory and Data Memory share the same Bank Pointer Register, care must be taken during programming.

BP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PMBP0	—	—	—	DMBP1	DMBP0
R/W	—	—	R/W	—	—	—	R/W	R/W
POR	—	—	0	—	—	—	0	0

Bit 7 ~ 6	Unimplemented, read as “0”
Bit 5	**PMBP0**: Program Memory Bank Pointer 0: Bank 0, Program Memory Address is from 0000H ~ 1FFFH 1: Bank 1, Program Memory Address is from 2000H ~ 3FFFH
Bit 4 ~ 2	Unimplemented, read as “0”
Bit 1 ~ 0	**DMBP1 ~ DMBP0**: Data Memory Bank Pointer 00: Bank 0 (for general purpose) 01: Bank 1 (for LCD) 10: Bank 2 (for sine generator) 11: Bank 3 (for general purpose)

Status Register – STATUS

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

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

The Z, OV, C, AC flags generally reflect the statuses of the latest operations.

In addition, on entering the interrupt sequence or executing the subroutine call, the status register will not be pushed onto stack automatically. If the contents of status are important and the subroutine can corrupt the status register, the programmer has to take precautions to save it properly.

STATUS Register

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

"x" unknown

bit 7~6 Unimplemented, read as "0"

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

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

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

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

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

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

System Control Registers – CTRL0, CTRL1

These registers are used to provide control over various internal functions. Some of these include the Timer/Event Counter 2 internal source option, the LCD driver clock (fSUB) option, Power down mode clock control, certain system clock options, external Interrupt edge trigger type, and the 32.768kHz crystal oscillator (LXT) enable control.

CTRL0 Register

Bit	7	6	5	4	3	2	1	0
Name	TCKS1	TCKS0	LFS	LCDCK1	LCDCK0	FSUBC	FSUBS	LXTLP
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	1

Bit 7~6 **TCKS1~TCKS0**: Timer/Event Counter 2 internal source selection
00: f_L (Low frequency clock)
01: f_{REF} (Reference frequency clock generated from Touch Key module)
10: f_{SEN} (Sensor frequency clock generated from Touch Key module)
11: f_{TMCK} (Gated Sensor frequency clock generated from Touch Key module)
If the Touch Key module is disabled, the TCKS1 and TCKS0 bits are always set to “00” and can not be written to.

Bit 5 **LFS**: Low Frequency clock source f_L selection
0: LIRC oscillator
1: LXT oscillator

Bit 4~3 **LCDCK1~LCDCK0**: To select the LCD driver clock
00: LCD clock = $f_{SUB}/3$
01: LCD clock = $f_{SUB}/4$
10: LCD clock = $f_{SUB}/8$
11: LCD clock = $f_{SUB}/8$

Bit2 **FSUBC**: f_{SUB} Power down mode clock control
0: disabled
1: enabled

Bit1 **FSUBS**: f_{SUB} Clock source selection
0: LIRC oscillator
1: LXT Oscillator

Bit0 **LXTLP**: LXT oscillator low power control function
0: LXT Oscillator quick start-up mode
1: LXT Oscillator Low Power Mode

CTRL1 Register

Bit	7	6	5	4	3	2	1	0
Name	EINTC1	EINTC0	—	—	—	—	BZCS	LXTEN
R/W	R/W	R/W	—	—	—	—	R/W	R/W
POR	0	0	—	—	—	—	0	1

Bit 7~6 **EINTC1, EINTC0**: External interrupt edge selection
00: disable
01: falling edge trigger
10: rising edge trigger
11: dual edge trigger

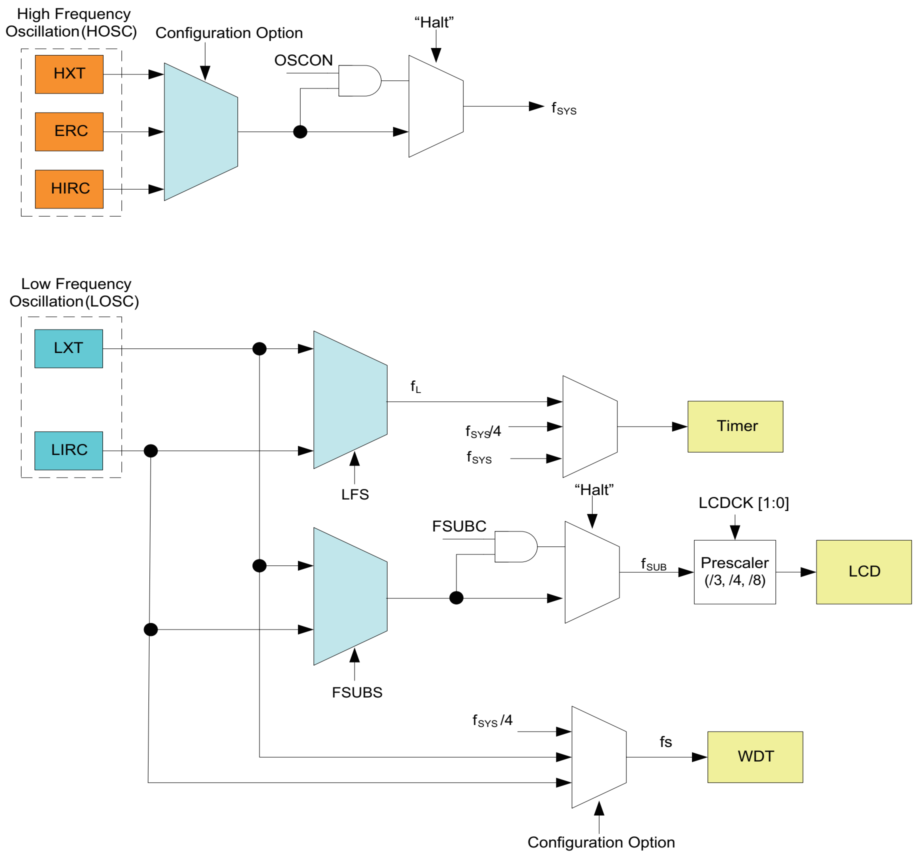
Bit 5~2 Unimplemented, read as “0”

Bit 1 **BZCS**: Buzzer clock source selection
0: from Timer/Event Counter 0
1: from Timer/Event Counter 1

Bit 0 **LXTEN**: LXT oscillator control in Power down mode
0: disabled
1: enabled

Oscillator Configuration

The device provides three system oscillator circuits known as a crystal oscillator (HXT), an external RC oscillator (ERC) and an internal high speed RC oscillator (HIRC) which are used for the system clock. There are also an internal 12kHz RC (LIRC) and a 32.768kHz crystal oscillator (LXT) which can provide a source clock for the WDT clock named f_S, the LCD driver clock, named f_{SUB} and the Timer/Event counters low frequency clock, named f_L ,for various timing purposes.



System Clock Configurations

In the Power down mode, the system oscillator, the internal 12kHz RC oscillator (LIRC) or the external 32.768kHz crystal oscillator (LXT) may be enabled or disabled depending upon the corresponding clock control bit described in the relevant sections. The system can be woken-up from the Power down mode by the occurrence of an interrupt, a transition determined by configuration options on any of the Port A pins, a WDT overflow or a timer overflow. The accompanying table illustrates the Oscillator type list.

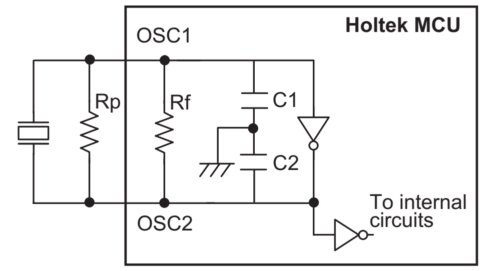
Type	Name	Freq.	Pins
External Crystal	HXT	400kHz~12MHz	OSC1/OSC2
External RC	ERC	400kHz~12MHz	OSC1
Internal High Speed RC	HIRC	4, 8 or 12MHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
Internal Low Speed RC	LIRC	12kHz	—

Oscillator Types

External Crystal/Ceramic Oscillator – HXT

The External Crystal/Ceramic System Oscillator is one of the system oscillator choices, which is selected via configuration options. For most crystal oscillator configurations, the simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, without requiring external capacitors and resistors. However, if a resonator instead of crystal is connected between OSC1 and OSC2, to ensure oscillation, it may be necessary to add two small value capacitors, C1 and C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.

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



Note: 1. Rp is normally not required.
3. Although not shown pins have a parasitic capacitance of around 7pF.

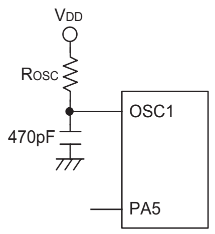
External Crystal/ Ceramic Oscillator

External RC Oscillator – ERC

Using the ERC oscillator only requires that a resistor, with a value between 24kΩand 1.5MΩ, is connected between OSC1 and VDD, and a capacitor is connected between OSC1 and ground, providing a low cost oscillator configuration. It is only the external resistor that determines the oscillation frequency; the external capacitor has no influence over the frequency and is connected for stability purposes only. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations

on the oscillation frequency are minimised. As a resistance/ frequency reference point, it can be noted that with an external 120KΩ resistor connected and with a 5V voltage power supply and temperature of 25 ℃ degrees, the oscillator will have a frequency of 4MHz within a tolerance of 2%. Here only the OSC1 pin is used, which is shared with I/O pin PA6, leaving pin PA5 free for use as a normal I/O pin.

For oscillator stability and to minimise the effects of noise and crosstalk, it is important to locate the capacitor and resistoras close to the MCU as possible.



External RC Oscillator – ERC

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of either 4MHz, 8MHz or 12MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of either 3V or 5V and at a temperature of 25℃ degrees, the fixed oscillation frequency of 4MHz, 8MHz or 12MHz will have a tolerance within 2%. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PA5 and PA6 are free for use as normal I/O pins.

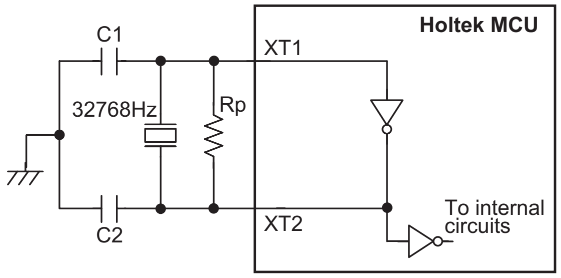
External 32.768kHz Crystal Oscillator – LXT

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

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

The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer’s specification. The external parallel feedback resistor, Rp, is required.

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



Note: 1. RTC OSC: without build-in RC
2. Rp, C1 and C2 are required.
3. Although not shown pins have a parasitic capacitance of around 7pF.

External 32.768kHz Crystal Oscillator - LXT

LXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
32768Hz	8pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. R_P = 5M~10MΩ is recommended.		

32768Hz Crystal Recommended Capacitor Values

LXT Oscillator Low Power Function

The LXT oscillator can function in one of two modes, the Quick Start Mode and the Low Power Mode. The mode selection is executed using the LXTLP bit in the CTRL0 register.

LXTLP Bit	LXT Mode
0	Quick Start
1	Low-power

After power on, the LXTLP bit will be automatically cleared to zero ensuring that the LXT oscillator is in the Quick Start operating mode. In the Quick Start Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up it can be placed into the Low-power mode by setting the LXTLP bit high. The oscillator will continue to run but with reduced current consumption, as the higher current consumption is only required during the LXT oscillator start-up. In power sensitive applications, such as battery applications, where power consumption must be kept to a minimum, it is therefore recommended that the application program sets the LXTLP bit high about 2 seconds after power-on.

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

Internal 12kHz Oscillator – LIRC

The Internal 12kHz RC Oscillator is one of the low frequency oscillator choices, which is selected via configuration option. It is a fully integrated RC oscillator with a typical period of approximately 65us at 5V, requiring no external components for its implementation. If the system enters the Power Down Mode, the internal RC oscillator can still continue to run if its clock is necessary to be used to clock the functions for timing purpose such as the WDT function, LCD Driver or Timer/Event Counters. The internal RC oscillator can be disabled only when it is not used as the clock source for all the peripheral functions determined by the configuration options of the WDT function and the relevant control bits which determine the clock is enabled or disabled for related peripheral functions.

Watchdog Timer – WDT

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 WDT is implemented using an internal 12kHz RC oscillator known as LIRC, the external LXT 32.768kHz oscillator or the instruction clock which is the system clock divided by 4.

Watchdog Timer Operation

The timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. A configuration option determines whether the Watchdog Timer is to be always on or whether its enable/disable control is under software control. If the configuration option chooses the always on option, then any WE4~WE0 values other than 01010 or 10101 will result in an MCU reset being generated. If the configuration option chooses the software control option then the Watchdog Timer can be disabled by setting the WE4~WE0 bits to a 10101 value. A value of 01010 will enable the Watchdog Timer and any other value will generate an MCU reset. The actual reset will be generated after 2~3 LIRC clock cycles.

WDT Config. Option	WE4 ~ WE0 bits	WDT Function
Always On	01010	Enabled
	10101	Enabled
	Other values	MCU reset generated after 2~3 LIRC clock cycles
Software Control	01010	Enabled
	10101	Disabled
	Other values	MCU reset generated after 2~3 LIRC clock cycles

WDT Functional Control Summary

The application can generate a software reset by writing a value other than 01010 and 10101 into the WE4~WE0 bits. If this is done then the WRF flag in the WDTC1 register will be set.

The WDT clock is divided by an internal counter to give a ratio division with a range of 2^8~2^{15} selected using the WS0~WS2 bits, to give a longer watchdog time-out period.

If the watchdog timer is disabled, the WDT timer will not generate a chip reset. So in the watchdog timer disable mode, the WDT timer counter can be read out and can be cleared. This function is used for the application program to access the WDT frequency to get the temperature coefficient for analog component adjustment. The LIRC oscillator can be disabled or enabled by the oscillator enable control bits LIRCEN1 and LIRCEN0 in the WDT control register WDTC for power saving reasons.

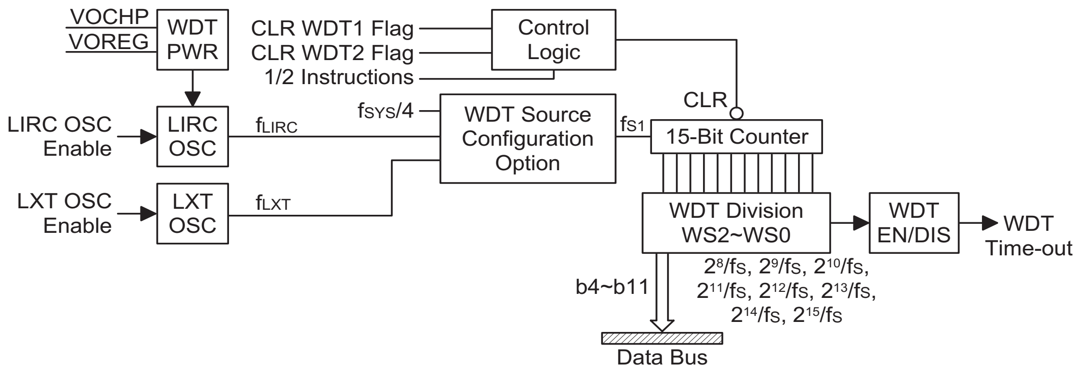
There are three registers related to the WDT function named WDTC, WDTC1 and WDTD. The WDTC and WDTC1 registers control the WDT oscillator enable/disable and the WDT power source. The WDTD register is the WDT counter content register and is read only.

The WDT power source selection bits named LIRCPWR1 and LIRCPWR0 are used to choose the WDT power source. The WDT default power source is from VOCHP. The main purpose of the regulator is to be used for WDT Temperature-coefficient adjustment. In this case, the application program should enable the regulator before switching to the Regulator source. The LIRCEN1 and LIRCEN0 bits can be used to enable or disable the LIRC oscillator. If the application does not use the LIRC oscillator, then it needs to disable it in order to save power. When the LIRC oscillator is disabled, then it is actually turned off, regardless of the setting of the relevant control bits which select the LIRC oscillator as its clock source. When the LIRC oscillator is enabled, it can be used as the clock source in the Power Down mode defined by the corresponding control bits of the peripheral functions.

The WDT clock source may also come from the instruction clock, in which case the WDT will operate in the same manner except that in the Power Down mode the WDT may stop counting and lose its protecting purpose. If the device operates in a noisy environment, using the on-chip LIRC oscillator is strongly recommended, since the HALT instruction will stop the system clock.

When the WDT overflows under normal operation a device reset will be executed and the status bit "TO" will be set. In the Power down mode, the overflow executes a warm reset, here only the PC and SP bits are reset to zero. There are three methods to clear the contents of the WDT, an external reset - a low level on RES -, a software instruction or a HALT instruction.

There are two types of software instructions; the single "CLR WDT" instruction, or the pair of instructions, "CLR WDT1" and "CLR WDT2". Of these two types of instruction, only one type of instruction can be active at a time depending on the configuration option – "CLR WDT" times selection option. If the "CLR WDT" is selected (i.e., CLR WDT times equal one), any execution of the "CLR WDT" instruction clears the WDT. If the "CLR WDT1 and CLR WDT2" option is chosen (i.e., CLR WDT times equal two), these two instructions have to be executed to clear the WDT, otherwise the WDT may reset the device due to a time-out.



Watchdog Timer

WDTC Register

Bit	7	6	5	4	3	2	1	0
Name	WS2	WS1	WS0	—	LIRCEN1	LIRCEN0	LIRCPWR1	LIRCPWR0
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	s	s	0	1

Bit 7~5 **WS2~WS0**: WDT prescaler rate select
000: $2^8/f_S$
001: $2^9/f_S$
010: $2^{10}/f_S$
011: $2^{11}/f_S$
100: $2^{12}/f_S$
101: $2^{13}/f_S$
110: $2^{14}/f_S$
111: $2^{15}/f_S$

Bit 4 unimplemented, read as "0"

Bit 3~2 **LIRCEN1~ LIRCEN0**: LIRC oscillator enable/disable control bits
00: LIRC oscillator is enabled
01: LIRC oscillator is disabled
10: LIRC oscillator is enabled
11: LIRC oscillator is enabled
It is strongly recommended to use “10” for WDT OSC enable

Bit 1~0 **LIRCPWR1~ LIRCPWR0**: LIRC Power Source Select
00: WDT power comes from VOCHP
01: WDT power comes from VOCHP
10: WDT power comes from Regulator
11: WDT power comes from VOCHP
It is strongly recommended to use “01” for VOCHP to prevent the noise to let the WDT lose the power
The WDT clock (f_S) is further divided by an internal counter to give longer watchdog time-out period. In this device, the division ratio can be varied by selecting different values of WS2~WS0 bits to give $2^8/f_S$ to $2^{15}/f_S$ division ratio range.

WDTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	WE4	WE3	WE2	WE1	WE0	LVRF	LRF	WRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	x	0	0

Bit 7 ~ 3 **WE4~WE0**: WDT enable/disable/MCU reset control
01010B: enable – power on value
10101B: Function depends upon WDT configuration option.
For the WDT always on option this value will enable the WDT.
For the software control option this value will disable the WDT.
Other values: Generates MCU reset after 2~3 LIRC clock cycles.

Bit 2 **LVRF**: LVRF reset flag
0: LVR not active
1: LVR active
This bit can be cleared to "0" by the application program, but can not be set to "1".

Bit 1 **LRF**: reset caused by LVRC setting
0: not active
1: active
This bit can be cleared to "0" by the application program, but can not be set to "1".

Bit 0 **WRF**: reset generated by WE4~WE0 bits

WDTD Register

Bit	7	6	5	4	3	2	1	0
Name	WDTD7	WDTD6	WDTD5	WDTD4	WDTD3	WDTD2	WDTD1	WDTD0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **WDTD7~ WDTD0**: WDT Counter value (bit 4~bit 11)
This register is read only and used for temperature adjusting

Operating Modes

The device has two operational modes, the Normal mode and the Power Down mode. In the Normal mode, the high speed system clock may come from external RC (ERC), external crystal (HXT) or internal RC (HIRC) oscillator. When in the Power down mode, the clocks in this device are all enabled or disabled using software. The accompanying table illustrates the operating modes and system clock control.

Operating Mode Control

HALT Instruction	Mode	System Oscillator	FSUBC	f_{SUB} Clock	LXTEN	LXT Oscillator (XT1/XT2)
Not executed	Normal	On	x	Enable	x	On
Executed	Power Down	On(OSCON=1) Off(OSCON=0)	0	Disable	1	On
	Power Down	On(OSCON=1) Off(OSCON=0)	1	Enable	1	On
	Power Down	On(OSCON=1) Off(OSCON=0)	0	Disable	0	Off
	Power Down	On(OSCON=1) Off(OSCON=0)	1	Enable	0	Off

Note: The LIRCEN1 and LIRCEN0 bits in the WDTC register can be setup to enable the LIRC. If the LVR is enabled, then the LIRC will also be enabled. However, as the LIRC is powered by the internal Voltage Regulator, which is controlled by the REGEN bit in the CHPRC register, then when the REGEN bit is cleared to zero, the LIRC and the LVR will both be disabled. Therefore, the REGEN bit should be set "1" to enable these two functions. Refer to the WDT section for the details regarding the WDTOSC setup.

Power Down Mode

The Power down mode is initialised by the "HALT" instruction and results in the following.

- The system oscillator stops running if the system oscillator is selected to be turned off by clearing the OSCON bit in the HALTC register to zero. Otherwise, the system oscillator will keep running if it is selected to be turned on in the power down mode.
- The contents of the Data Memory and of the registers remain unchanged.
- The WDT is cleared and starts recounting if the WDT clock source is from the LIRC or the LXT oscillator.
- All I/O ports maintain their original status.
- The PDF flag is set but the TO flag is cleared.
- The LCD driver keeps running if the LCD clock f_{SUB} is enabled by setting the FSUBC bit high and the LCDON bit in the HALTC register is set high.

The system leaves the Power down mode by means of an external reset, an interrupt, an external transition signal on Port A, or a WDT overflow. An external reset causes a device initialisation, and the WDT overflow performs a "warm reset". After examining the TO and PDF flags, the reason for the device reset can be determined. The PDF flag is cleared by system power-up or by executing the "CLR WDT" instruction, and is set by executing the "HALT" instruction. On the other hand, the TO flag is set if WDT time-out occurs, and causes a wake-up that only resets the program counter and SP, and leaves the others in their original state.

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each pin of port A can be independently selected to wake-up the device using configuration options. After awakening from an I/O port stimulus, the program will resume execution at the next instruction. However, if awakening from an interrupt, two sequences may occur. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program will resume execution at the next instruction. But if the interrupt is enabled, and the stack is not full, the regular interrupt response takes place.

When an interrupt request flag is set before entering the "HALT" status, the system cannot be awakened using that interrupt.

If a wake-up events occur, it takes a number of clock cycles to resume normal operation. In other words, a dummy period is inserted after the wake-up. If the wake-up results from an interrupt acknowledgment, the actual interrupt subroutine execution is delayed by more than one cycle. However, if the wake-up results in the next instruction execution, the execution will be performed immediately after the dummy period is finished. To minimise power consumption, all the I/O pins should be carefully managed before entering the HALT status.

The accompanying table illustrates the wake-up delay time for different system clock sources.The SST time is decided by SST configuration option and the OSCON bit in the HALTC register.

Fsys Clock Source	SST Selection	OSCON	SST Time (N: number of fsys clock)
XTAL	0	0	N= 1024
	0	1	N= 2
	1	0	N= 1024
	1	1	N= 2
ERC	0	0	N= 1024
	0	1	N= 2
	1	0	N= 2
	1	1	N= 2
HIRC	0	0	N= 1024
	0	1	N= 2
	1	0	N= 2
	1	1	N= 2

Required Wake-up Clock Cycles

HALTC Register

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

bit 7 **OSCON**: System oscillator state in the Power down mode
0: System oscillator stops running
1: System oscillator keeps running

bit 6~1 unimplemented, read as "0"

bit 0 **LCDON**: LCD module state in the Power down mode
0: LCD state is determined by the LCD_ON configuration option
1: LCD module remains on (if the f_{SUB} is active) regardless of the configuration option setting

Reset and Initialisation

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

One of these registers is the Program Counter, which will be reset to zero forcing the microcontroller to begin program execution from the lowest Program Memory address. In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high.

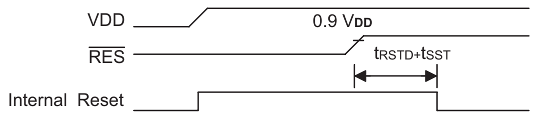
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset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

Reset Functions

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

Power-on Reset

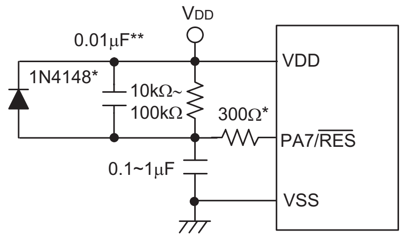
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions.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



Note: t_{RSTD} is power-on delay, typical time=100ms

Power-On Reset Timing Chart

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



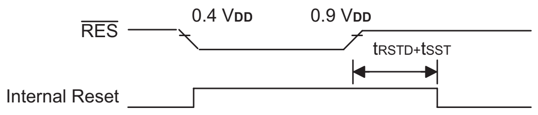
Note: "*" It is recommended that this component is added ESD protection.
"**" It is recommended that this component is added in environments where power line noise is significant.

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

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

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

This type of reset occurs when the microcontroller is already running and the $\overline{\text{RES}}$ pin is forcefully pulled low by external hardware such as an external switch. In this case as in the case of other reset, the Program Counter will reset to zero and program execution initiated from this point.



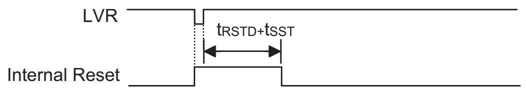
Note: t_{RSTD} is power-on delay, typical time= 100ms

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

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function has a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the WDTC1 register will also be set to 1. The LVR includes the following specifications: For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for greater than the value t_{LVR} specified in the A.C. characteristics. If the low voltage state does not exceed t_{LVR}, the LVR will ignore it and will not perform a reset function. One of a range of specified voltage values for V_{LVR} can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits have any other value, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the device after 2~3 LIRC clock cycles. When this happens, the LRF bit in the WDTC1 register will be set to 1. After power on the register will have the value of 01010101B.

Note that the LVR function will be automatically disabled when the device enters the power down mode.



Note: t_{RSTD} is power-on delay, typical time= 100ms

Low Voltage Reset Timing Chart

- **LVRC Register**

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

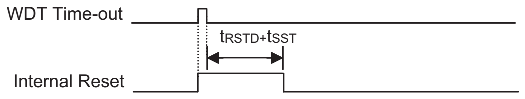
Bit 7~0 **LVS7 ~ LVS0**: LVR voltage select
01010101: 2.1V (default)
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Other values: Generates MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage values above, an MCU reset will be generated. The reset operation will be activated after 2~3 LIRC clock cycles. In this situation the register contents will remain the same after such a reset occurs.

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

Watchdog Time-out Reset during Normal Operation

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

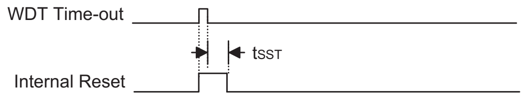


Note: t_{RSTD} is power-on delay, typical time= 100ms

WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during Power Down mode

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



Note: The t_{SST} can be chosen to be either 1024 or 2 clock cycles via configuration option if the system clock source is provided by ERC or HIRC. The SST is 1024 for HXT or LXT.

WDT Time-out Reset during Power down 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 Power down function or Watchdog Timer. The reset flags are shown in the table.

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	RES or LVR reset during Normal operation
0	1	RES Wake-up HALT
1	u	WDT time-out reset during Normal operation
1	1	WDT Wake-up HALT

Note: "u" stands for unchanged.

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

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT	Clear after reset, WDT begins counting
Timer/Event Counter	Timer Counter will be turned off
Prescaler	The Timer Counter Prescaler will be cleared
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	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)*
IAR0	---- ----	---- ----	---- ----	---- ----	---- ----
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
IAR1	---- ----	---- ----	---- ----	---- ----	---- ----
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	---0 --00	---0 --00	---0 --00	---0 --00	---u --uu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	---- ----	---- ----	---- ----	---- ----	---- ----
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
CTRL0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
LVRC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PBC	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)*
DACO	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
FTRC	0--0 --00	0--0 --00	0--0 --00	0--0 --00	u--u --uu
ADCR	-000 x000	-000 x000	-000 x000	-000 x000	-uuu uuuu
SWC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADCD	0000 -111	0000 -111	0000 -111	0000 -111	uuuu -uuu
WDTC1	0101 0x11	0101 0x11	0101 0x11	0101 0x11	uuuu uuuu
WDTC	111- ss01	111- ss01	111- ss01	111- ss01	uuu- uuuu
WDTD	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu –uuu
CHPRC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR2H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu
SGC	0--0 ----	0--0 ----	0--0 ----	0--0 ----	u--u ----
SGDNR	---0 0000	---0 0000	---0 0000	---0 0000	---u uuuu
SGN	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
HALTC	0--- ---0	0--- ---0	0--- ---0	0--- ---0	u--- ---u
LCDOUT	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
CTRL1	00-- --01	00-- --01	00-- --01	00-- --01	uu-- --uu
VIBRC	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
IADAC	0-00 0000	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAWK	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
PBPU	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
PCPU	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
SFS	---- --00	---- --00	---- --00	---- --00	---- --uu
OPAC	0--- 0000	0--- 0000	0--- 0000	0--- 0000	u--- uuuu
LCDC	00-0 -000	00-0 -000	00-0 -000	00-0 -000	uu-u -uuu
MFIC	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
IAC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAC1	1111 00-1	1111 00-1	1111 00-1	1111 00-1	uuuu uu-u
TKM0C4	-000 ---0	-000 ---0	-000 ---0	-000 ---0	-uuu ---u
TKM016DH	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM016DL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM010DL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0RO	0000 0001	0000 0001	0000 0001	0000 0001	uuuu uuuu
TKM0C0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0C1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0C2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0C3	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

Note : *stands for "warm reset"　　"-" not implement
"u" stands for "unchanged"　　"x" stands for "unknown"
"s" for special case, it depends on the option table (please see the WDT chapter for the detail)

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. Most pins can have either an input or output designation under user program control. Additionally, as there are pull-high resistors and wake-up software configurations, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A, [m]", where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

I/O Register List

Register Name	POR	Bit							
		7	6	5	4	3	2	1	0
PAWK	00H	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
PAC	FFH	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	00H	—	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PBC	3FH	—	—	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	00H	—	—	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PCC	FFH	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	FFH	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selectable via a register known as PAPU, PBPU and PCPU located in the Data Memory. The pull-high resistors are implemented using weak PMOS transistors. Note that pin PA7 does not have a pull-high resistor selection.

PAPU Register

Bit	7	6	5	4	3	2	1	0
Name	—	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as “0”

Bit 6~0 **PAPU**: I/O Port bit 6 ~ bit 0 Pull-High Control
0: Disable
1: Enable

PBPU Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5~0 **PBPU**: I/O Port bit5 ~ bit 0 Pull-High Control
0: Disable
1: Enable

PCPU Register

Bit	7	6	5	4	3	2	1	0
Name	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PCPU**: I/O Port bit 7 ~ bit 0 Pull-High Control
0: Disable
1: Enable

Port A Wake-up

If the HALT instruction is executed, the device will enter Power down mode, where the system clock will stop resulting in power being conserved, 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 PA0~PA7 pins from high to low. After a HALT instruction forces the microcontroller into entering Power down mode, the processor will remain in a low-power state until the logic condition of the selected wake-up pin on Port A changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Note that pins PA0 to PA7 can be selected individually to have this wake-up feature using an internal register known as PAWK, located in the Data Memory.

PAWU Register

Bit	7	6	5	4	3	2	1	0
Name	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
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 **PAWU**: Port A bit 7 ~ bit 0 Wake-up Control
0: Disable
1: Enable

I/O Port Control Registers

Each Port has its own control register, known as PAC, PBC and PCC which controls the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

PAC Register

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

Bit 7~0 **PAC**: I/O Port bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

PBC Register

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

Bit 7~6 "—" Unimplemented, read as "0"

Bit 5~0 **PBC**: I/O Port bit 5 ~ bit 0 Input/Output Control
0: Output
1: Input

PCC Register

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

Bit 7~0 **PCC**: I/O Port bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

Pin-shared Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For some pins, the chosen function of the multi-function I/O pins is set by configuration options while for others the function is set by application program control.

External Interrupt Input

The external interrupt pin, INT, is pin-shared with an I/O pin. To use the pin as an external interrupt input the correct bits in the INTC0 register must be programmed. The pin must also be setup as an input by setting the PBC4 bit in the Port Control Register. A pull-high resistor can also be selected via the appropriate port pull-high resistor register. Note that even if the pin is setup as an external interrupt input the I/O function still remains.

External Timer/Event Counter Input

The Timer/Event Counter pins, TMR0, TMR1 and TMR2 are pin-shared with I/O pins. For these shared pins to be used as Timer/Event Counter inputs, the Timer/Event Counter must be configured to be in the Event Counter or Pulse Width Measurement Mode. This is achieved by setting the appropriate bits in the Timer/Event Counter Control Register. The pins must also be setup as inputs by setting the appropriate bit in the Port Control Register. Pull-high resistor options can also be selected using the port pull-high resistor registers. Note that even if the pin is setup as an external timer input the I/O function still remains.

Buzzer Output

The buzzer function output is pin-shared with an I/O pin. The output function of this pin is chosen using the SFS register. Note that the corresponding bit of the port control register, must setup the pin as an output to enable buzzer output. If the port control register has setup the pin as an input, then the pin will function as a normal logic input with the usual pull-high selection, even if buzzer function has been selected.

LCD Driver Pins

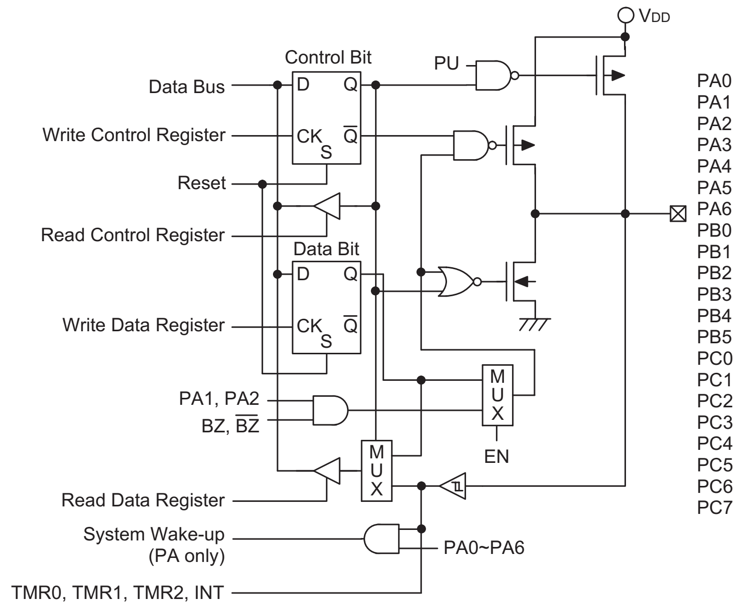
Pins PC0~PC7 on Port C can be used as LCD SEG driver pins. This function is controlled using the LCDOUT register.

Touch Key Pins

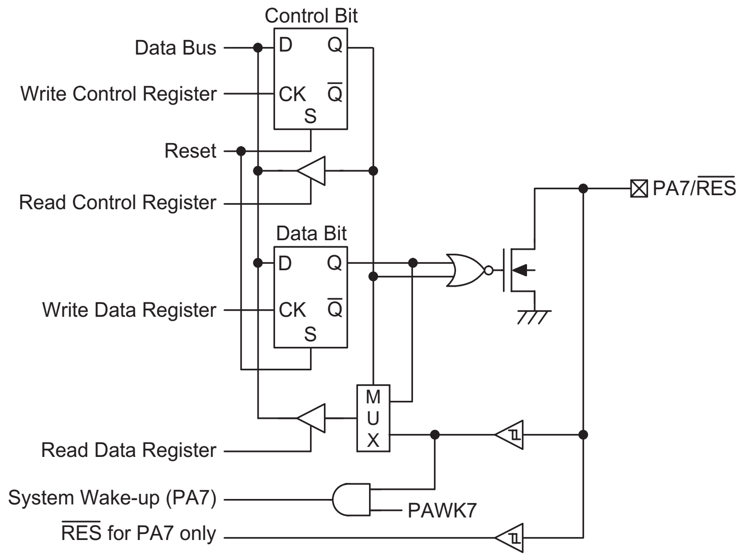
Pins PB0~PB3 are pin-shared with touch key function. This function is controlled using the TKM0C2 register.

I/O Pin Structures

The diagrams illustrate the I/O pin internal structures. As the exact logical construction of the I/O pin may differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins.



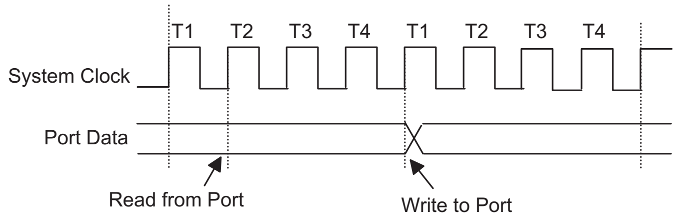
Generic Input/Output Ports



PA7 NMOS Input/Output Port

Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, the I/O data register and I/O port control register 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 options have been selected. 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 register is first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct value into the 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.



Read Modify Write Timing

Pins PA0 to PA7 each have a wake-up function, selected via the PAWK register. When the device is in the Idle/Sleep Mode, various methods are available to wake the device up. One of these is a high to low transition of any of these pins. Single or multiple pins on Port A can be setup to have this function.

Timer/Event Counters

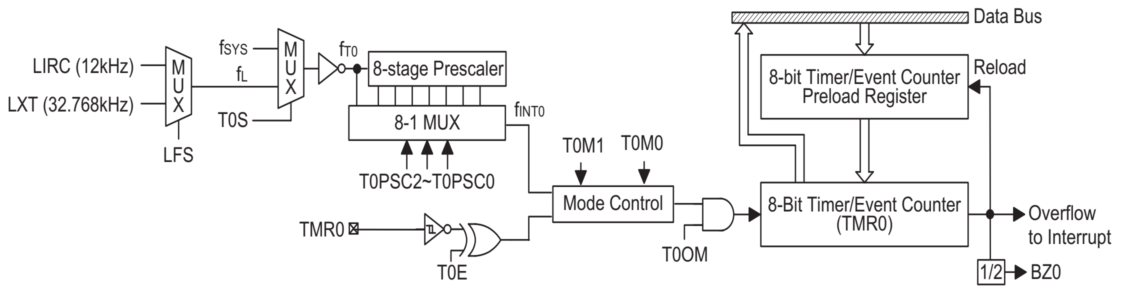
Three timer/event counters are implemented in the microcontroller. Timer/Event Counter 0 contains an 8-bit programmable count-up counter whose clock may come from an external source or an internal clock source. An internal clock source comes from f_{SYS} or the internal low frequency clock known as fL. Timer/Event Counter 1 contains a 16-bit programmable count-up counter whose clock may come from an external source or an internal clock source. An internal clock source comes from $f_{SYS}/4$ or the internal low frequency clock known as fL. The clock fL is derived from the LIRC or LXT oscillator and can be selected by the Low Frequency selection bit LFS bit in the CTRL0 register. The external clock input allows the user to count external events, measure time intervals or pulse widths, or to generate an accurate time base. Timer/Event Counter 2 contains a 16-bit programmable count-up counter whose clock may come from an external source or an internal clock source. An internal clock source comes from $f_{SYS}/4$ or the Timer/Event Counter 2 internal clock f_{TCK}. The clock f_{TCK} may come from the low frequency clock f_L or the clocks generated from the Touch Key module named f_{REF}, fSEN and f_{TMCK} described in the Touch Key Function section. The clock is selected using the Timer/Event Counter 2 clock source selection bits TCKS1 and TCK0 in the CTRL0 register. The external clock input allows the user to count external events, measure time intervals or pulse widths, or to generate an accurate time base.

There are two registers related to the Timer/Event Counter 0; TMR0 and TMR0C.Writing to TMR0 puts the starting value in the Timer/Event Counter 0 register and reading TMR0 reads out the contents of Timer/Event Counter 0. The TMR0C is a timer/event counter control register, which defines the overall operations. There are three registers related to the Timer/Event Counter 1; TMR1H, TMR1L and TMR1C. Writing to TMR1L will only put the written data into an internal lower-order byte buffer (8-bit) while writing to TMR1H will transfer the specified data and the contents of the lower-order byte buffer to both the TMR1H and TMR1L registers, respectively.

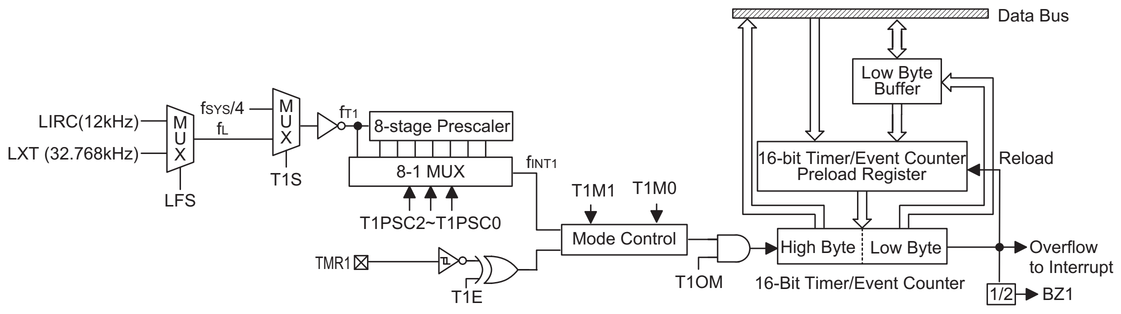
The Timer/Event Counter 1 preload register is changed when each time there is a write operation to TMR1H. Reading TMR1H will latch the contents of TMR1H and TMR1L counters to the destination and the lower-order byte buffer, respectively. Reading TMR1L will read the contents of the lower-order byte buffer. TMR1C is the Timer/Event Counter 1 control register, which defines the operating mode, counting enable or disable, the TMR1 active edge and the prescaler stage selections. Also there are three registers related to the Timer/Event Counter 2 named TMR2H, TMR2L and TMR2C. The operations of reading from and writing to the Timer/Event Counter 2 registers named TMR2H and TMR2L are the same with Timer/Event Counter 1 described above.

The TxM0 and TxM1 bits in TMRxC register where x may be equal to 0, 1 or 2 define the operation mode. The event count mode is used to count external events, which means that the clock source must come from the external (TMR0, TMR1 or TMR2) pin. The timer mode functions as a normal timer with the clock source coming from the internal selected clock source.

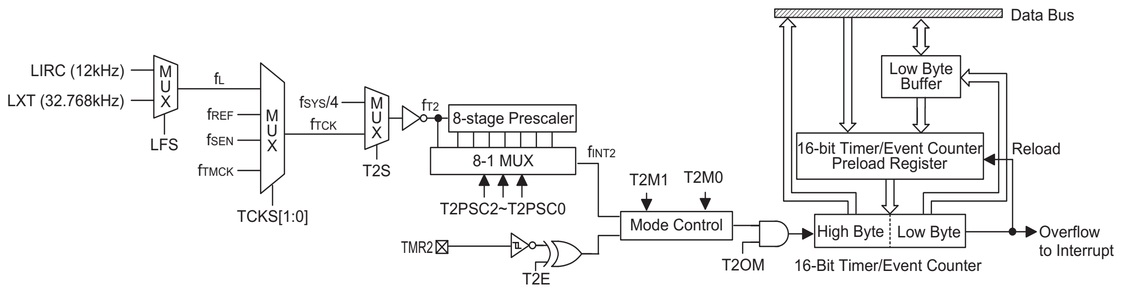
Finally, the pulse width measurement mode can be used to count a high or low level duration of an external signal on the TMR0, TMR1 or TMR2 pins with the timing based on the internally selected clock source. In the event count or timer mode, the Timer/Event Counter starts counting at the current contents in the Timer/Event Counter and ends at FFH for -8-bit counter or FFFFH for 16-bit counter. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag, T0F, T1F or T2F. In the pulse width measurement mode with the values of the Timer enable control bit TxON and the active edge control bit TxE equal to “1”, after the TMRx pin has received a transient from low to high (or high to low if the TxE bit is “0”), it will start counting until the TMRx pin returns to the original level and resets the TxON bit. The measured result remains in the timer/event counter even if the activated transient occurs again. Therefore, only a 1-cycle measurement can be made until the TxON bit is again set. The cycle measurement will re-function as long as it receives further transient pulses. In this operation mode, the timer/event counter begins counting not according to the logic level but to the transient edges. In the case of counter overflows, the counter is reloaded from the timer/event counter register and issues an interrupt request, as in the other two modes, i.e., event and timer modes.



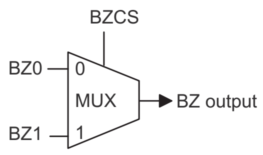
Timer/Event Count 0



Timer/Event Count 1



Timer/Event Count 2



TMR0C Register

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	T0S	T0ON	T0E	T0PSC2	T0PSC1	T0PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7~6 **T0M1, T0M0**: Timer0 operation mode selection
00: no mode available
01: event counter mode
10: timer mode
11: pulse width measurement mode

Bit 5 **T0S**: timer clock source
0: f_{SYS}
1: Low Frequency clock f_L

Bit 4 **T0ON**: Timer/event counter counting enable
0: disable
1: enable

Bit 3 **T0E**:
Event Counter active edge selection
0: count on rising edge
1: count on falling edge
Pulse Width measurement active edge selection
0: start counting on falling edge, stop on rising edge
1: start counting on rising edge, stop on falling edge

Bit 2~0 **T0PSC2, T0PSC1, T0PSC0**: Timer prescaler rate selection
Timer internal clock (f_{INT0})=
000: f_{T0}
001: $f_{T0}/2$
010: $f_{T0}/4$
011: $f_{T0}/8$
100: $f_{T0}/16$
101: $f_{T0}/32$
110: $f_{T0}/64$
111: $f_{T0}/128$

TMR1C Register

Bit	7	6	5	4	3	2	1	0
Name	T1M1	T1M0	T1S	T1ON	T1E	T1PSC2	T1PSC1	T1PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7~6 **T1M1, T1M0**: Timer0 operation mode selection
00: no mode available
01: event counter mode
10: timer mode
11: pulse width measurement mode

Bit 5 **T1S**: timer clock source
0: $f_{SYS}/4$
1: Low Frequency clock f_L

Bit 4 **T1ON**: Timer/event counter counting enable
0: disable
1: enable

Bit 3 **T1E**:
Event Counter active edge selection
0: count on rising edge
1: count on falling edge
Pulse Width measurement active edge selection
0: start counting on falling edge, stop on rising edge
1: start counting on rising edge, stop on falling edge

Bit 2~0 **T1PSC2, T1PSC1, T1PSC0**: Timer prescaler rate selection
Timer internal clock (f_{INT1})=
000: f_{T1}
001: $f_{T1}/2$
010: $f_{T1}/4$
011: $f_{T1}/8$
100: $f_{T1}/16$
101: $f_{T1}/32$
110: $f_{T1}/64$
111: $f_{T1}/128$

TMR2C Register

Bit	7	6	5	4	3	2	1	0
Name	T2M1	T2M0	T2S	T2ON	T2E	T2PSC2	T2PSC1	T2PSC0
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 **T2M1, T2M0**: Timer0 operation mode selection
- 00: no mode available
- 01: event counter mode
- 10: timer mode
- 11: pulse width measurement mode

Bit 5 **T2S**: Timer clock source
- 0: $f_{SYS}/4$
- 1: f_{TCK}

Bit 4 **T2ON**: Timer/event counter counting enable
- 0: disable
- 1: enable

Bit 3 **T2E**:

Event Counter active edge selection
- 0: count on rising edge
- 1: count on falling edge

Pulse Width Measurement active edge selection
- 0: start counting on falling edge, stop on rising edge
- 1: start counting on rising edge, stop on falling edge

Bit 2~0 **T2PSC2, T2PSC1, T2PSC0**: Timer prescaler rate selection

Timer internal clock (f_{INT2})=
- 000: f_{T2}
- 001: $f_{T2}/2$
- 010: $f_{T2}/4$
- 011: $f_{T2}/8$
- 100: $f_{T2}/16$
- 101: $f_{T2}/32$
- 110: $f_{T2}/64$
- 111: $f_{T2}/128$

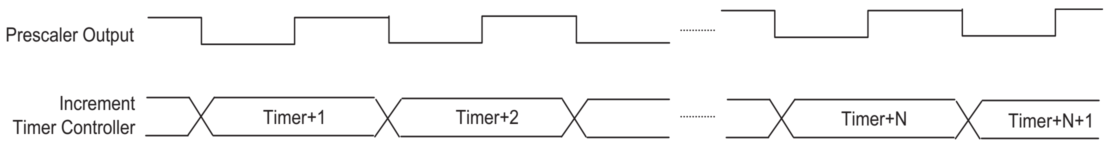
Timer Mode

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. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Timer Mode

Bit7	Bit6
1	0

In this mode the internal clock is used as the timer clock. The timer input clock source is either f_{SYS}, $f_{SYS}/4$, f_L or f_{TCK}. However, this timer clock source is further divided by a prescaler, the value of which is determined by the bits TnPSC2~TnPSC0 in the Timer Control Register. The timer-on bit, TnON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one; when the timer is full and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupt is one of the wake-up sources, however, the internal interrupts can be disabled by ensuring that the ETnI bits of the INTC0 and MFIC register are reset to zero.



Timer Mode Timing Chart

Event Counter Mode

In this mode, a number of externally changing logic events, occurring on the external timer TMRn pin, can be recorded by the Timer/Event Counter. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

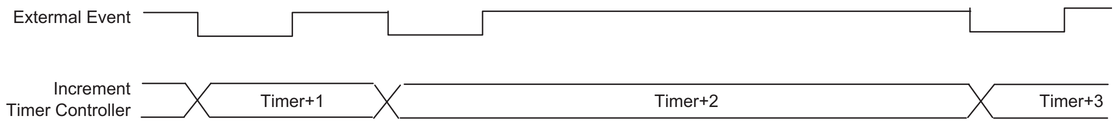
Control Register Operating Mode Select Bits for the Event Counter Mode

Bit7	Bit6
0	1

In this mode, the external timer TMRn pin is used as the Timer/Event Counter clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit TnON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. If the Active Edge Select bit, TnE, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the TnE is high, the counter will increment each time the external timer 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. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register is reset to zero.

As the external timer pin is shared with an I/O pin, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Event Counting

Mode, the second is to ensure that the port control register configures the pin as an input. It should be noted that in the event counting mode, even if the is in the Power Down Mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input TMRn pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (TnE=1)

Pulse Width Measurement Mode

In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Pulse Width Capture Mode

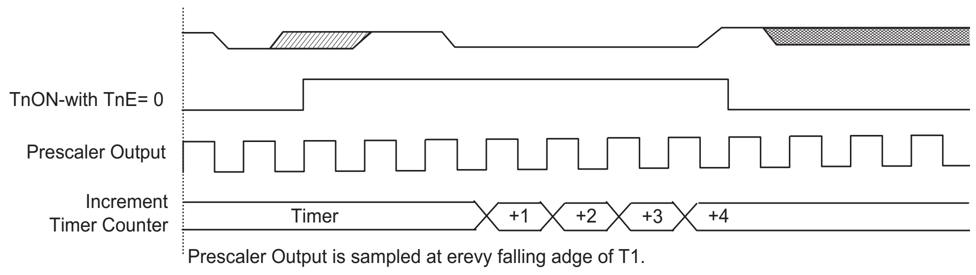
Bit7	Bit6
1	1

In this mode the internal clock, f_{SYS}, $f_{SYS}/4$, f_L or f_{TCK}, is used as the internal clock for the 8-bit Timer/Event Counter and the 16-bit Timer/Event Counter. However, the clock source, f_{SYS} and $f_{SYS}/4$, for the 8-bit timer and the 16-bit timer are further divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits TnPSC2~TnPSC0, which are bits 2~0 in the Timer Control Register. After the other bits in the Timer Control Register have been setup, the enable bit TnON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the external timer pin.

If the Active Edge Select bit TnE, which is bit 3 of the Timer Control Register, is low, once a high to low transition has been received on the external timer pin, the Timer/Event Counter will start counting until the external timer pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. If the Active Edge Select bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. It is important to note that in the pulse width measurement Mode, the enable bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under 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 TMRn 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 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. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register, is reset to zero.

As the TMRn pin is shared with an I/O pin, to ensure that the pin is configured to operate as a pulse width measurement pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the pulse width measurement mode, the second is to ensure that the port control register configures the pin as an input.



Pulse Width Measurement Mode Timing Chart (TnE=0)

Prescaler

Bits TnPSC0~TnPSC2 of the TMRnC register can be used to define a division ratio for the internal clock source of the Timer/Event Counter enabling longer time out periods to be setup.

Buzzer

Operating in a similar way to the Programmable Frequency Divider, the Buzzer function provides a means of producing a variable frequency output, suitable for applications such as Piezo-buzzer driving or other external circuits that require a precise frequency generator. The BZ and $\overline{BZ}$ pins form a complimentary pair, and are pin-shared with I/O pins, PA1 and PA2. Note that the $\overline{BZ}$ pin is the inverse of the BZ pin which together generates a differential output which can supply more power to connected interfaces such as buzzers. The PA1 and PA2 are setup as I/O pins or buzzer pins using the SFS register.

The buzzer is driven by the Timer/Event Counter 0 or Timer/Event Counter 1 overflow signal divided by 2 selected by the clock source selection bit named BZCS in CTRL1 register.

If the software options have selected both pins PA1 and PA2 to function as a BZ and $\overline{BZ}$ complementary pair of buzzer outputs, then for correct buzzer operation it is essential that both pins must be setup as outputs by setting bits PAC1 and PAC2 of the PAC port control register to zero. The PA1 data bit in the PA data register must also be set high to enable the buzzer outputs, if set low, both pins PA1 and PA2 will remain low. In this way the single bit PA1 of the PA data register can be used as an on/off control for both the BZ and $\overline{BZ}$ buzzer pin outputs. Note that the PA2 data bit in the PA data register has no control over the BZ buzzer pin PA2.

If software options have selected that only the PA1 pin is to function as a BZ buzzer pin, then the PA2 pin can be used as a normal I/O pin. For the PA1 pin to function as a BZ buzzer pin, PA1 must be setup as an output by setting bit PAC1 of the PAC port control register to zero. The PA1 data bit in the PA data register must also be set high to enable the buzzer output, if set low pin PA1 will remain low. In this way the PA1 bit can be used as an on/off control for the BZ buzzer pin PA1. If the PAC1 bit of the PAC port control register is set high, then pin PA1 can still be used as an input even though the Software option has configured it as a BZ buzzer output.

BZBS	BZS	PAC2	PAC1	PA2	PA1	Function
1	1	0	0	X	1	PA1= BZ, PA2= $\overline{\text{BZ}}$
1	1	0	0	X	0	PA1= "0", PA2= "0"
1	1	0	1	X	1	PA1= "Input", PA2= $\overline{\text{BZ}}$
1	1	0	1	X	0	PA1= "Input", PA2= "0"
1	1	1	0	X	1	PA1= BZ, PA2= "Input"
1	1	1	0	X	0	PA1= "0", PA2= "Input"
X	X	1	1	X	X	PA1= "Input", PA2= "Input"
0	0	X	X	X	X	PA1= "I/O", PA2= "I/O"

"×" stands for don't care

SFS Register

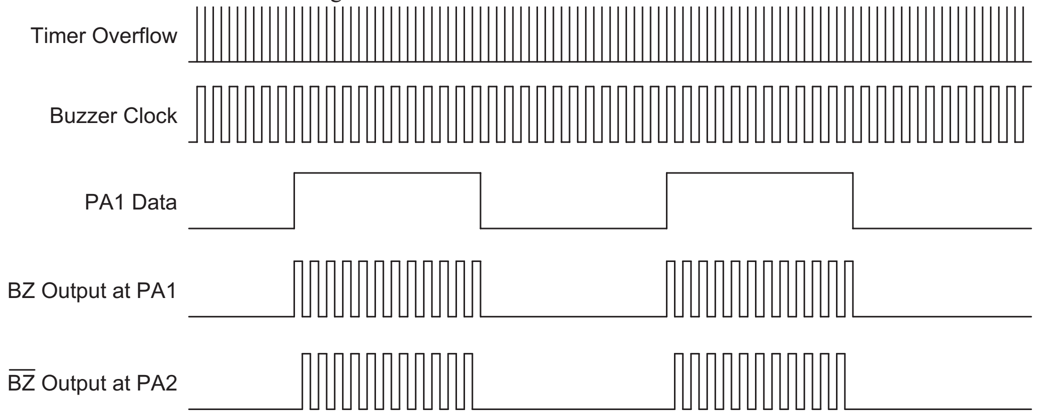
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	BZBS	BZS
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1 **BZBS**: PA2 function selection
0: I/O
1: $\overline{\text{BZ}}$

Bit 0 **BZS**: PA1 function selection
0: I/O
1: BZ

When register "SFS" selects the I/O function, the related control refers to the Port A control register.



Buzzer Output Pin Control

I/O Interfacing

The Timer/Event Counter, when configured to run in the event counter or pulse width measurement mode, requires the use of an external timer pin for its operation. As this pin is a shared pin it must be configured correctly to ensure that it is setup for use as a Timer/Event Counter input pin. This is achieved by ensuring that the mode select bits in the Timer/Event Counter control register, select either the event counter or pulse width measurement mode. Additionally the corresponding Port Control Register bit must be set high to ensure that the pin is setup as an input. Any pull-high resistor connected to this pin will remain valid even if the pin is used as a Timer/Event Counter input.

Programming Considerations

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

When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, this 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 enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The edge select, timer mode and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application. It is also important to ensure that an initial value is first loaded into the timer registers before the timer is switched on; this is because after power-on the initial values of the timer registers are unknown. After the timer has been initialised the timer can be turned on and off by controlling the enable bit in the timer control register.

When the Timer/Event Counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the Timer/Event Counter interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in a Power-down condition. This situation may occur if the Timer/Event Counter is in the Event Counting Mode and if the external signal continues to change state. In such a case, the Timer/Event Counter will continue to count these external events and if an overflow occurs the device will be woken up from its Power-down condition. 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.

Timer Program Example

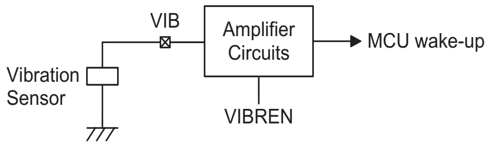
The program shows how the Timer/Event Counter registers are setup along with how the interrupts are enabled and managed. Note how the Timer/Event Counter is turned on, by setting bit 4 of the Timer Control Register. The Timer/Event Counter can be turned off in a similar way by clearing the same bit. This example program sets the Timer/Event Counters to be in the timer mode, which uses the internal system clock as their clock source.

PFD Programming Example

```
org  04h                    ;external interrupt vector
org  0ch                    ;Timer Counter 0 interrupt vector
jmp  tmr0int                ;jump here when Timer 0 overflows
:           :
org  20h                    ;main program
:           :
                            ;internal Timer 0 interrupt routine
tmr0int:
:           :
                            ;Timer 0 main program placed here
:           :
begin:
                            ;setup Timer 0 registers
mov  a,09bh                 ;setup Timer 0 preload value
mov  tmr0,a
mov  a,081h                 ;setup Timer 0 control register
mov  tmr0c,a                ;timer mode and prescaler set to /2
                            ;setup interrupt register
mov  a,005h                 ;enable master interrupt and both timer interrupts
mov  intc0,a
:           :
set  tmr0c.4                ;start Timer 0
:           :
```

Vibration Sensor Amplifier

The device contains a Vibration Sensor Amplifier to amplify small signal inputs, generated from vibration sensors. When the sensor is connected to the vibration input pin, namely VIB, and a small signal resulting from vibration detection is generated on the VIB pin, the internal amplifier will amplify the signal to generate a wake-up source to wake up the device from the power down mode. The Vibration Sensor Amplifier can be enabled or disable by the control bit, VIBREN, in the VIBRC register to save power.



VIBRC Register

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **VIBREN**: Vibration Sensor Amplifier control
0: disabled
1: enabled

Touch Key Module

The device contains four touch keys. The touch key functions are fully integrated and require no external components, allowing touch key functions to be implemented by the simple manipulation of internal registers.

Touch Key Structure

The touch keys are pin shared with the PB logic I/O pins, with the desired function chosen via register bits.

Touch Key Register Definition

The touch key functions have their own suite of registers. The following table shows the register set together with a basic description.

Name	Usage
TKM016DH	16-bit C/F counter high byte
TKM016DL	16-bit C/F counter low byte
TKM010DL	10-bit counter low byte
TKM0RO	Internal Capacitor Select
TKM0C0	Control Register 0 Key Select/X2 freq/filter control/frequency select
TKM0C1	Control Register 1 Sensor Oscillator Control/Touch key or I/O select.
TKM0C2	Control Register 2 Counter on-off and clear control/reference clock control/Start bit
TKM0C3	Control Register 3 Counter overflow bits/Reference Oscillator Overflow Time Select
TKM0C4	Control Register 4 f_{VDD} Clock Source Select/Power control

Register Listing

Register Name	Bits							
	7	6	5	4	3	2	1	0
TKM016DH	D7	D6	D5	D4	D3	D2	D1	D0
TKMn16DL	D7	D6	D5	D4	D3	D2	D1	D0
TKM010DL	D7	D6	D5	D4	D3	D2	D1	D0
TKM0RO	D7	D6	D5	D4	D3	D2	D1	D0
TKM0C0	M0MXS1	M0MXS0	M0DFEN	M0FILEN	M0SOFC	M0SOF2	M0SOF1	M0SOF0
TKM0C1	M0K4OEN	M0K3OEN	M0K2OEN	M0K1OEN	M0K4IO	M0K3IO	M0K2IO	M0K1IO
TKM0C2	M016CTON	M010CTON	M0ST	M0ROEN	M0RCCLR	M016CTCLR	M010CTCLR	M0ROS
TKM0C3	D9	D8	M0RCOV	M016CTOV	M010CTOV	M0ROVS2	M0ROVS1	M0ROVS0
TKM0C4	—	FVDD2	FVDD1	FVDD0	—	—	—	NORM

Register Content Summary

TKM016DH Register

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

Bit 7~0 **D15~D8**: Touch key module 16-bit counter (TMCNT) high byte.

TKM016DL Register

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

Bit 7~0 **D7~D0**: Touch key module 16-bit counter (TMCNT) low byte.

TKM010DL Register

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

Bit 7~0 **D7~D0**: touch key module 10-bit counter (TMCNT) bit 7~ bit 0.

TKM0RO 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	1

Bit 7~0 **D7~D0**: Integrated OSC capacitor can select TKMORCC[7:0]×50pF/256. It is not permitted to write a 00H value to this register.

TKM0C0 Register

Bit	7	6	5	4	3	2	1	0
Name	M0MXS1	M0MXS0	M0DFEN	M0FILEN	M0SOFC	M0SOF2	M0SOF1	M0SOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **M0MXS1~M0MXS0**: multiplexer key select.
00: KEY 1
01: KEY 2
10: KEY 3
11: KEY 4

Bit 5 **M0DFEN**: double frequency function control
0: disable
1: enable

Bit 4 **M0FILEN**: filter function control
0: disable
1: enable

Bit 3 **M0SOFC**: C to F OSC frequency hopping function select
0: frequency hopping is controlled by software and C to F OSC micro adjustment frequency is determined by the M0SOF2~M0SOF0 bits
1: frequency hopping controlled by hardware - M0SOF2~M0SOF0 bits have no effect.
Here the counter's three MSBs, selected by M0ROVS2~M0ROVS0, will automatically adjust the C to F OSC micro adjustment frequency.

Bit 2~0 **M0SOF2~ M0SOF0**: C to F OSC frequency hopping frequency select
000: 400 kHz
001: 425 kHz
010: 450 kHz
011: 475 kHz
100: 500 kHz
101: 525 kHz
110: 550 kHz
111: 575 kHz

TKM0C1 Register

Bit	7	6	5	4	3	2	1	0
Name	M0K4OEN	M0K3OEN	M0K2OEN	M0K1OEN	M0K4IO	M0K3IO	M0K2IO	M0K1IO
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 **M0K4OEN**: KEY 4 Sensor Oscillator Control
0: disable
1: enable

Bit 6 **M0K3OEN**: KEY 3 Sensor Oscillator Control
0: disable
1: enable

Bit 5 **M0K2OEN**: KEY 2 Sensor Oscillator Control
0: disable
1: enable

Bit 4 **M0K1OEN**: KEY 1 Sensor Oscillator Control
0: disable
1: enable

Bit 3 **M0K4IO**: KEY 4 touch key input or I/O function
0: I/O
1: Touch key input

Bit 2 **M0K3IO**: KEY 3 touch key input or I/O function
0: I/O
1: Touch key input

Bit 1 **M0K2IO**: KEY 2 touch key input or I/O function
0: I/O
1: Touch key input

Bit 0 **M0K1IO**: KEY 1 touch key input or I/O function
0: I/O
1: Touch key input

TKM0C2 Register

Bit	7	6	5	4	3	2	1	0
Name	M016CTON	M010CTON	M0ST	M0ROEN	M0RCCLR	M016CTCLR	M010CTCLR	M0ROS
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 **M016CTON**: 16-bit C/F Counter Control
0: disable
1: enable
This bit is cleared by hardware when the M0RCOV bit is set to 1.

Bit 6 **M010CTON**: 10-bit Counter Control
0: disable
1: enable
This bit is cleared by hardware when the M0RCOV bit is set to 1.

Bit 5 **M0ST**: Start f_{TMCK} and f_{VDDCK} output clock
0: Hardware sets ENCK to low, reference oscillator 13-stage counter stops
0→1: Enable the f_{TMCK} and f_{VDDCK} output clock - hardware sets ENCK to high. Hardware sets clears the M0RCOV flag low and the Reference Oscillator 13-stage counter start counting. If the counter overflows, the M0RCOV flag is set to “1”, the hardware clears ENCK to low and the Touch Key Interrupt request flag, TKF, will be set. A Touch Key Interrupt will be generated if the Touch Key Interrupt enable bit, TKE, and the Global Interrupt bit, EMI, are set to 1.

Bit 4 **M0ROEN**: Reference Clock Control
0: disable
1: enable

Bit 3 **M0RCCLR**: Clear Reference Oscillator 13-stage counter
0: no change
1: clear counter
The hardware circuit will clear this bit to zero after it is set to “1” by the user.

Bit 2 **M016CTCLR**: 16-bit C/F Counter Clear Control
0: no change
1: clear counter
Hardware will clear this bit to zero after it is set to “1” by user.

Bit 1 **M010CTCLR**: 10-bit counter Clear Control
0: no change
1: clear counter
Hardware will clear this bit to zero after it is set to “1” by user.

Bit 0 **M0ROS**: Reference Oscillator 13-stage Counter Clock Source Select
0: Reference OSC
1: KEY 4 Sensor OSC

TKM0C3 Register

Bit	7	6	5	4	3	2	1	0
Name	D9	D8	M0RCOV	M016CTOV	M010CTOV	M0ROVS2	M0ROVS1	M0ROVS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **D9~ D8**: fVDD 10-bit Counter Bit 9 and Bit 8

Bit 5 **M0RCOV**: 13-stage Counter Overflow Flag
0: no overflow
1: overflow
If the 13-stage counter overflows, the Touch Key Interrupt request flag, TKF, will be set and the hardware will clear ENCK to low to disable the fTMCK and fVDDCK outputs.

Bit 4 **M016CTOV**: 16-bit C/F Counter Overflow Flag
0: no overflow
1: overflow
This bit must be cleared by software.

Bit 3 **M010CTOV**: 10-bit Counter Overflow Flag
0: no overflow
1: overflow
This bit must be cleared by software.

Bit 2~0 **M0ROVS2~ M0ROVS0**: Reference Oscillator 13-stage Counter Overflow Time Select Switch 0 control bits:
000: 64 counts
001: 128 counts
010: 256 counts
011: 512 counts
100: 1024 counts
101: 2048 counts
110: 4096 counts
111: 8192 counts

TKM0C4 Register

Bit	7	6	5	4	3	2	1	0
Name	—	FVDD2	FVDD1	FVDD0	—	—	—	NORM
R/W	—	R/W	R/W	R/W	—	—	—	R/W
POR	—	0	0	0	—	—	—	0

Bit 7 Unimplemented, read as "0"

Bit 6~4 **FVDD2~ FVDD0**: FVDD Clock Source Select
000: $f_{SYS}/2$
001: $f_{SYS}/4$
010: $f_{SYS}/8$
011: $f_{SYS}/16$
100: $f_{SYS}/32$
101: $f_{SYS}/64$
110: $f_{SYS}/128$
111: f_{SUB}

Bit 3~1 Unimplemented, read as "0"

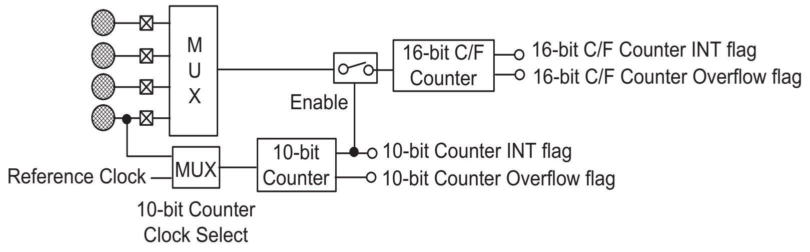
Bit 0 **NORM**: Normal/Low Power Mode Select
0: Low power mode
1: Normal mode

Touch Key Operation

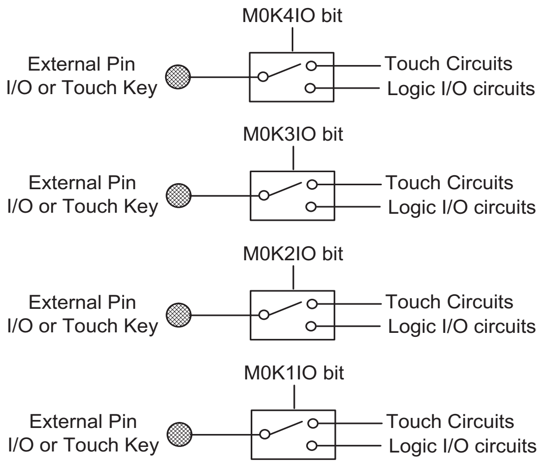
When a finger touches or is in proximity to a touch pad, the capacitance of the pad will increase. By using this capacitance variation to change slightly the frequency of the internal sense oscillator, touch actions can be sensed by measuring these frequency changes. Using an internal programmable divider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

The device contains four touch key inputs which are shared with logical I/O pins, with the desired function selected using register bits. The Touch Key module also has its own interrupt vectors and set of interrupts flags.

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


M U X
16-bit C/F Counter
16-bit C/F Counter INT flag
16-bit C/F Counter Overflow flag
Enable
10-bit Counter
10-bit Counter INT flag
10-bit Counter Overflow flag
Reference Clock
MUX
10-bit Counter Clock Select

Touch Switch Module Block Diagram



Touch Key or I/O Function Select

Touch Key Interrupts

The touch key module, which consists of four touch keys, has three independent interrupts, an overall touch key interrupt as well as one for the 16-bit C/F counter and one for the 10-bit counter. The 16-bit C/F counter and 10-bit counter interrupts are contained within the Multi-function interrupts and therefore do not have their own vectors. Care must be taken during programming as these counter interrupt flags contained within the Multi-function interrupts will not be automatically reset upon entry into the interrupt service routine but rather must be reset manually by the application program. More details regarding the touch key interrupts are located in the interrupt section of the datasheet.

Programming Considerations

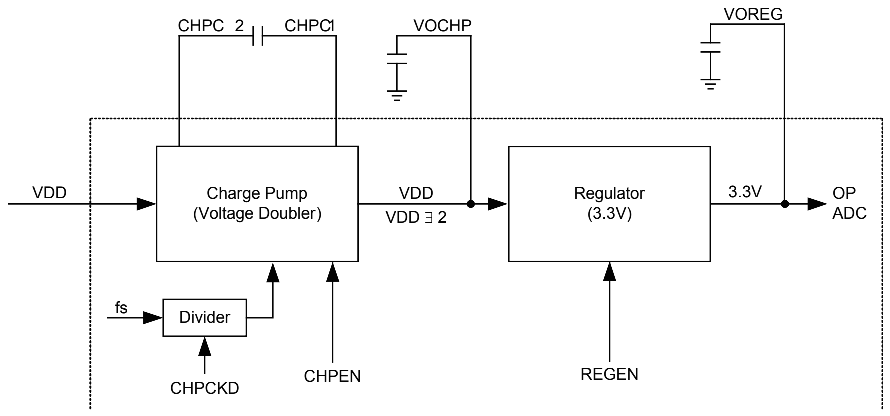
After the relevant registers are setup, the touch key detection process is initiated the changing the M0ST bit from low to high. This will enable and synchronise all relevant oscillators.

Charge Pump and Voltage Regulator

One charge pump and one voltage regulator are implemented in this device as way of providing a stable voltage source for certain internal functions. An additional Bandgap voltage source is also provided.

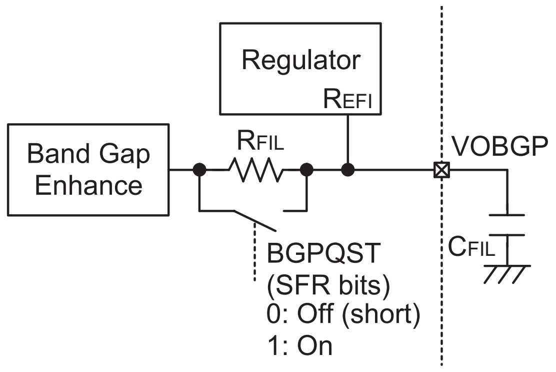
Operation

The charge pump can be enabled or disabled by the application program. The charge pump uses VDD as its input, and has the function of doubling the VDD voltage. The output voltage of the charge pump will be VDD×2. The regulator can generate a stable voltage of 3.3V, which is used by the internal WDT and A/D Converter and can also provide an external bridge sensor excitation voltage or supply a reference voltage for other applications. The user needs to guarantee that the charge pump output voltage is greater than 3.6V to ensure that the regulator generates the required 3.3V voltage output. The block diagram of this module is shown below.



Charge Pump and Regulator Block Diagram

Additionally, the device also includes a band gap voltage generator for the 1.5V low temperature sensitive reference voltage. This reference voltage is used as the zero adjustment and for a single end type reference voltage.



RFIL is about 100KΩ and the recommend CFIL is 10μF.

Note: The VOBGP signal is only for internal used. It must not be connected to external components except for the recommend CFIL capacitor.

There is a single register associated with this module named CHPRC. The CHPRC is the Charge Pump/Regulator Control register, which controls the charge pump on/off, regulator on/off functions as well as a clock divider value to generate the charge pump clock. The CHPCKD4~CHPCKD0 bits are use to set the clock divider to generate the desired clock frequency for proper charge pump operation. The actual frequency is determined by the following formula.

Actual Charge Pump Clock= (f_{SYS}/16) / (CHPCKD +1).

CHPRC Register

Bit	7	6	5	4	3	2	1	0
Name	CHPCKD4	CHPCKD3	CHPCKD2	CHPCKD1	CHPCKD0	BGPQST	CHPEN	REGEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~3 **CHPCKD4~CHPCKD0**: The Charge pump clock divider
These 5 bits select the clock divider ration of 1~32.
Charge Pump clock = (f_{SYS}/16) / (CHPCKD+1)

Bit 2 **BGPQST**: Bandgap quick start-up function
0: R short, quickly start
1: R connected, normal RC filter mode
Every time when REGEN changes from 0 to 1 (regulator turns on), this bit should be set to 0 and then set to 1 to make sure the Bandgap stabilises quickly (The minimum time is about 2ms now).

Bit 1 **CHPEN**: Charge Pump on/off control
0: disable
1: enable
Note: this bit will be ignored if the REGEN is disabled.

Bit 0 **REGEN**: Regulator/Charge-Pump module on/off control
0: disable
1: enable

REGEN	CHPEN	Charge Pump	VOCHP Pin	Regulator	VOREG Pin	OPA, ADC Body Fat Circuit	Description
0	×	OFF	VDD	OFF	Hi-Impedance	Disable	Whole module is disabled, OPA/ADC/body fat circuit will lose power
1	0	OFF	VDD	ON	3.3V	Active	Used when V_{DD} is greater than 3.6V (V_{DD}> 3.6V)
1	1	ON	2×VDD	ON	3.3V	Active	Use when V_{DD} is less than 3.6V (V_{DD}= 2.2V - 3.6V)

The suggested charge pump clock frequency is 20kHz. The application needs to set the correct value to get the desired clock frequency. For a 4MHz application, the CHPCKD bits should be set to the value 11, and for a 2MHz application, the bits should be set to 5.

The REGEN bit in the CHPRC register is the Regulator/Charge-pump module enable/disable control bit. If this bit is disabled, then the regulator will be disabled and the charge pump will be also be disabled to save power. When REGEN= 0, the module will enter the power down mode ignoring the CHPEN setting. The A/D Converter and OPA will also be disabled to reduce power.

If REGEN is set to "1", the regulator will be enabled. If CHPEN is enabled, the charge pump will be active and will use V_{DD} as its input to generate the double voltage output. This double voltage will be used as the input voltage for the regulator. If CHPEN is set to "0", the charge pump is disabled and the charge pump output will be equal to the charge pump input V_{DD}.

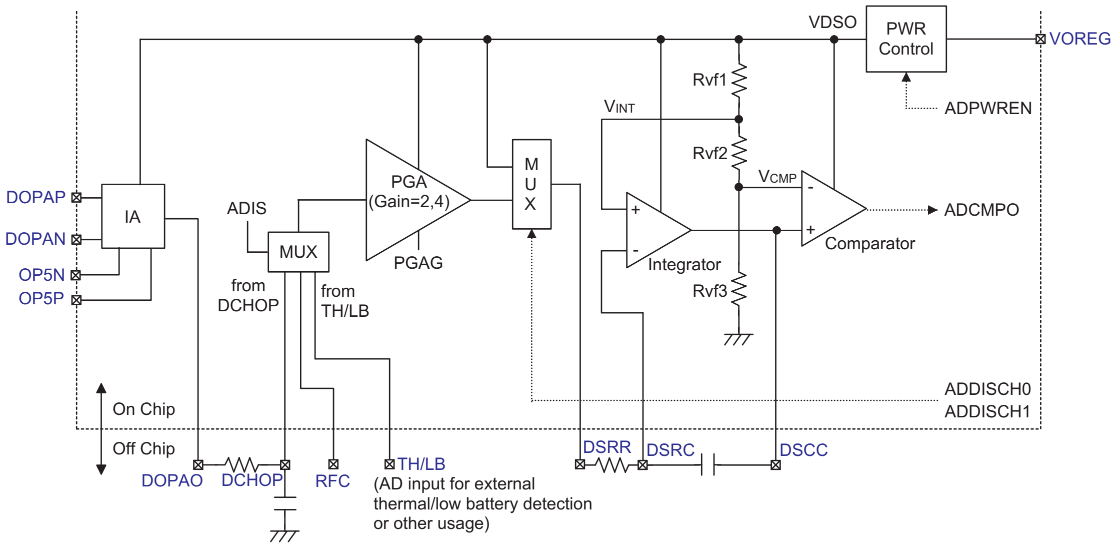
It is necessary to take care of the V_{DD} voltage. If the voltage is less than 3.6V, then CHPEN should be set to 1 to enable the charge pump, otherwise CHPEN should be set to zero. If the Charge pump is disabled and V_{DD} is less than 3.6V then the output voltage of the regulator will not be guaranteed.

Dual Slope A/D Converter

A Dual Slope A/D converter is implemented in the device. The dual slope module includes an Instrumentation Amplifier, a Programmable Gain Amplifier, for the amplification of differential signals, an Integrator and a comparator for the main dual slope AD convertor.

There are two special function registers related to this function known as ADCR and ADCD. The ADCR register is the A/D control register, which controls the ADC block power on/off, the chopper clock on/off, the charge/discharge control and is also used to read out the comparator output status. The ADCD register is the A/D Chopper clock divider register, which defines the chopper clock to the ADC module.

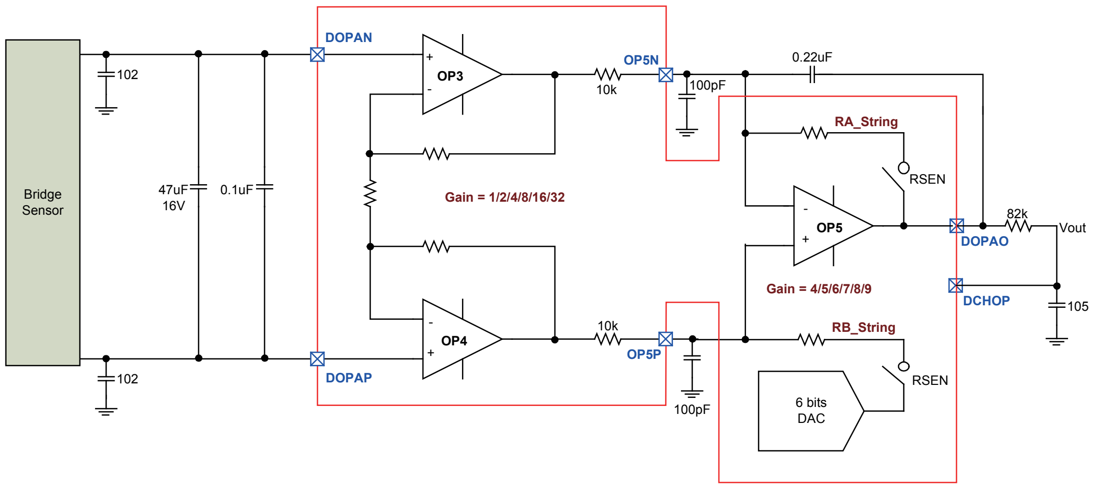
The ADPWREN bit, defined in ADCR register, is used to control the ADC module on/off function. The ADCCKEN bit, defined in the ADCR register, is used to control the chopper clock on/off function. When ADCCKEN is set to "1" it will enable the Chopper clock, with the clock frequency defined by the ADCD register. The ADC module includes the OPA, PGA, integrator and comparator. However, the Bandgap voltage generator is independent of this module. It will be automatically enabled when the regulator is enabled, and also be disabled when the regulator is disabled. The application program should enable the related power to permit them to function and disable them when entering the power down mode to conserve power. The charge/discharge control bits, ADDISCH1 and ADDISCH0, are used to control the Dual slope circuit charging and discharging behavior. The ADCMPO bit is read only for the comparator output, while the ADINTM bits can set the ADCMPO trigger mode for interrupt generation. The ADC PGA input signal can come from the DCHOP or TH/LB pin selected by the ADIS selection bit in ADCD register. The PGA gain can be either 2 or 4 determined by the PGAG gain selection bit in the ADCD register. The reference voltages of the ADC integrator and comparator named VINT and VCMP shown in the Dual Slope ADC structure diagram can be selected by the ADRR0 selection bit.



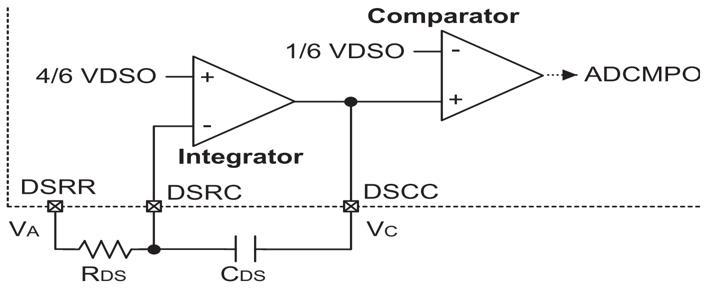
Note: V_{INT}, V_{CMP} signal can come from different R groups which are selected by software registers.

Dual Slope ADC Structure

An instrumentation amplifier is included to provide a high CMRR amplifying interface to differential signals from external sensors. With a standard instrumentation amplifier setup of three operational amplifiers and a DAC for offset calibration, this structure provides a low noise sensor interface. Two registers are used for overall control of the instrumentation amplifier and a single register for DAC control. If these functions are not used they should be disabled using the register bits to reduce power consumption.



Instrumentation Amplifier



Dual Slope Anolog Digital Convertor Operation

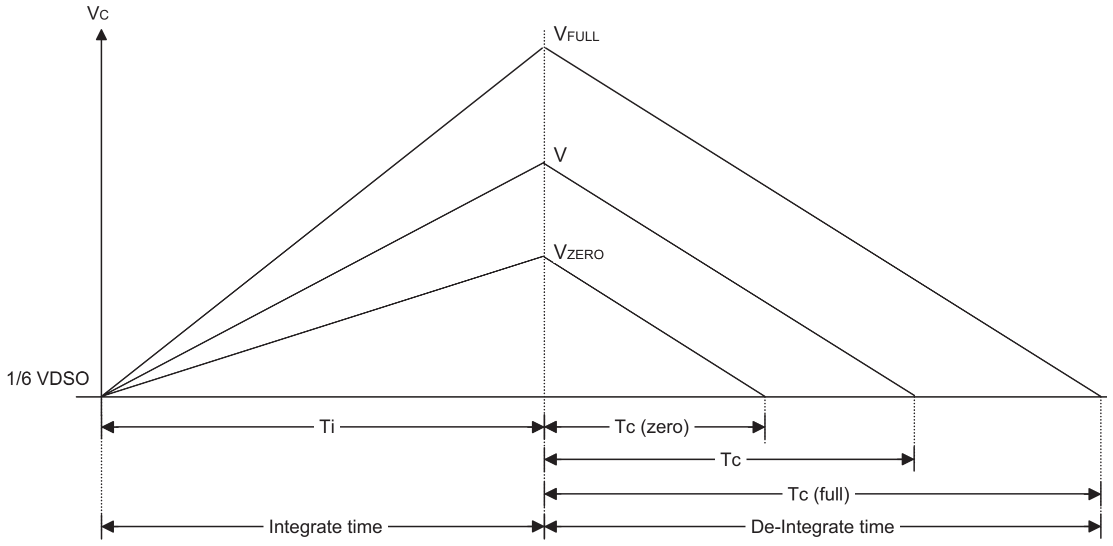
The following descriptions are based on the fact that the ADRR0 bit is set to "0". The instrumentation amplifier and PGA combination, form a differential input pre-amplifier which amplifies the sensor input signal. The combination of the Integrator, the comparator, the resistor R_{ds}, between DSRR and DSRC and the capacitor C_{ds}, between DSRC and DSCC, form the main body of the Dual slope ADC. The Integrator integrates the output voltage increase or decrease and is controlled by the "Switch Circuit" - refer to the block diagram. The integration and de-integration curves are illustrated by the following.

The "comparator" will switch the state from high to low when V_C, which is the DSCC pin voltage, drops to less than 1/6 VDSO.

In general applications, the application program will switch the ADC to the charging mode for a fixed time called T_i, which is the integrating time. It will then switch to the discharging mode and wait for V_C to drop to less than 1/6 VDSO. At this point the comparator will change state and store the time taken, T_C, which is the de-integrating time. The following formula 1 can then be used to calculate the input voltage V_A.

formula 1: $V_A = (1/3) \times VDSO \times (2 - T_C/T_i)$.
(Based on ADRR0=0)

In user applications, it is required to choose the correct value of R_{ds} and C_{ds} to determine the T_i value, to allow the V_C value to operate between 5/6 VDSO and 1/6 VDSO. VFULL cannot be greater than 5/6 VDSO and V_{ZERO} cannot be less than 1/6 VDSO.



ADCR Register

Bit	7	6	5	4	3	2	1	0
Name	—	ADCCKEN	ADINTM1	ADINTM0	ADCMPO	ADDISCH1	ADDISCH0	ADPWREN
R/W	—	R/W	R/W	R/W	R	R/W	R/W	R/W
POR	—	0	0	0	×	0	0	0

"×" unknown

Bit 7 Unimplemented, read as "0"

Bit 6 **ADCCKEN**: A/D Convertor OP chopper clock source on/off switching
0: disable
1: enable (clock value is defined by ADCD register)

Bit 5~4 **ADINTM1, ADINTM0**: ADCMPO data interrupt trigger mode definition
00: no interrupt
01: rising edge
10: falling edge
11: both edges

Bit 3 **ADCMPO**: Dual Slope ADC - last stage comparator output. Read only bit, write data instructions will be ignored. During the discharging state, when the integrator output is less than the reference voltage,the ADCMPO will change from high to low.

Bit 2~1 **ADDISCH1~ ADDISCH0**: ADC discharge/charge definition
00: reserved
01: charging (Integrator input connect to buffer output)
10: discharging (Integrator input connect to VDSO)
11: reserved

Bit 0 **ADPWREN**: Dual slope block (including input OP) power on/off switching
0: disable Power
1: Power source comes from the regulator

ADCD Register

Bit	7	6	5	4	3	2	1	0
Name	PGAG	ADIS1	ADIS0	ADRR0	—	ADCD2	ADCD1	ADCD0
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	0	0	0	0	—	1	1	1

Bit 7 **PGAG**: PGA gain selection
0: gain = 2
1: gain = 4

Bit 6~5 **ADIS1~ADIS0**: A/D PGA input selection
00: from RFC pin
01: from TH/LB pin
10: from DCHOP pin
11: reserved

Bit 4 **ADRR0**: A/D integrator and comparator reference voltage selection
0: (VINT, VCMP) = (4/6 VDSO, 1/6 VDSO)
1: (VINT, VCMP) = (4.4/6 VDSO, 1/6 VDSO)

Bit 3 Unimplemented, read as "0"

Bit 2~0 **ADCD2~ADCD0**: Chopper clock definition (ADCCKEN should be enable), the suggestion clock is around 10kHz
000: (f_{SYS}/32)/1
001: (f_{SYS}/32)/2
010: (f_{SYS}/32)/4
011: (f_{SYS}/32)/8
100: (f_{SYS}/32)/16
101: (f_{SYS}/32)/32
110: (f_{SYS}/32)/64
111: (f_{SYS}/32)/128

IAC0 Register

Bit	7	6	5	4	3	2	1	0
Name	IAEN	G12	G11	G10	IA2EN	G02	G01	G00
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 **IAEN**: OP3, OP4 on/off control
0: off
1: on

bit 6 ~ 4 **G12~G10**: Instrumentation Amplifier 2nd Stage Gain
000: 4
001: 5
010: 6
011: 7
100: 8
101: 9
110: undefined
111: undefined

bit 3 **IA2EN**: OP5 on/off control.
0: off
1: on

bit 2 ~ 0 **G02~G00**: Instrumentation Amplifier 1st Stage Gain
000: 1
001: 2
010: 4
011: 8
100: 16
1xx: 32

IAC1 Register

Bit	7	6	5	4	3	2	1	0
Name	CHOPEN	BI_PGA1	BI_PGA0	IAD2	IAD1	IAD0	—	RSEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	R/W
POR	1	1	1	1	0	0	0	1

'x' unknown

bit 7 **CHOPEN**: OP3, OP4 Chopper Enable
0: disable
1: enable

bit 6 ~ 5 **BI_PGA1~BI_PGA0**: OP3 and OP4 Operating Current Selection
00: undefined
01: undefined
10: undefined
11: 7I (op3 and op4 current=400µA, IA (op3, op4, op5) current=1.2mA)

bit 4 ~ 2 **IAD2~IAD1**: Chopper Clock Select
IA chopper clock defined as:
000: clock= $(f_{SYS}/32)/1$
001: clock= $(f_{SYS}/32)/2$
010: clock= $(f_{SYS}/32)/4$
011: clock= $(f_{SYS}/32)/8$
100: clock= $(f_{SYS}/32)/16$
101: clock= $(f_{SYS}/32)/32$
110: clock= $(f_{SYS}/32)/64$
111: clock= $(f_{SYS}/32)/128$
A suggested clock speed has a range of around 8-10kHz. Note that CHOPEN must be enabled – here the IA chopper clock is approximate 8k ~10k.(7.8125k ~ 11.71875k) and the ADC chopper clock must be < 8k.

bit 1 Unimplemented, read as "0"

bit 0 **RSEN**: OP5 external/internal gain setup control bit
0: Open loop setup – gain defined by external reisistors.
1: Closed loop setup – gain defined by G10, G11 and G12 bits in the IAC0 register.

IADAC Register

Bit	7	6	5	4	3	2	1	0
Name	IADAEN	—	D5	D4	D3	D2	D1	D0
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 **IADAEN**: IA DAC enable/disable
0: disable
1: enable

bit 6 Unimplemented, read as "0"

bit 5 ~ 0 **D5~D0**: DAC Output Voltage Setup Bits
Note: The DAC output range is 0V ~ $\{V_{OREG}-(V_{OREG}/64)\}$
1 LSB = 1 digital code = $V_{OREG}/64$
Output Voltage= $V_{OREG}/64*(D[5:0])$

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 an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs.

The device provides an external interrupt and multiple internal interrupts. The external interrupt is controlled by the action of the external interrupt pin, while the internal interrupts are controlled by the Timer/Event Counters, Touch Key and A/D converter.

Interrupt Register

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by using three registers, INTC0, INTC1 and MFIC. By controlling the appropriate enable bits in this registers each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request flag will be set by the microcontroller. The global enable flag if cleared to zero will disable all interrupts.

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	T1F	T0F	EIF	ET1I	ET0I	EEI	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 **T1F**: Timer/Event Counter 1 interrupt request flag
0: inactive
1: active

Bit 5 **T0F**: Timer/Event Counter 0 interrupt request flag
0: inactive
1: active

Bit 4 **EIF**: External interrupt request flag
0: inactive
1: active

Bit 3 **ET1I**: Timer/Event Counter 1 interrupt enable
0: disable
1: enable

Bit 2 **ET0I**: Timer/Event Counter 0 interrupt enable
0: disable
1: enable

Bit 1 **EEI**: External interrupt enable
0: disable
1: enable

Bit 0 **EMI**: Master interrupt global enable
0: disable
1: enable

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	TKF	ADF	MFF	—	TKE	EADI	EMFI
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 **TKF**: Touch key interrupt request flag
0: inactive
1: active

Bit 5 **ADF**: A/D Convertor request flag
0: inactive
1: active

Bit 4 **MFF**: Multi-function interrupt request flag
0: inactive
1: active

Bit 3 unimplemented, read as “0”

Bit 2 **TKE**: Touch key interrupt enable
0: disable
1: enable

Bit 1 **EADI**: A/D Convertor interrupt enable
0: disable
1: enable

Bit 0 **EMFI**: Multi-function interrupt enable
0: disable
1: enable

MFIC Register

Bit	7	6	5	4	3	2	1	0
Name	—	M16CTF	M10CTF	T2F	—	M16CTE	M10CTE	ET2I
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 **M16CTF**: Touch Key Module 16-bit Counter interrupt request flag
0: inactive
1: active

Bit 5 **M10CTF**: Touch Key Module 10-bit Counter interrupt request flag
0: inactive
1: active

Bit 4 **T2F**: Timer/Event Counter 2 interrupt request flag
0: inactive
1: active

Bit 3 unimplemented, read as “0”

Bit 2 **M16CTE**: Touch Key Module 16-bit Counter interrupt enable
0: disable
1: enable

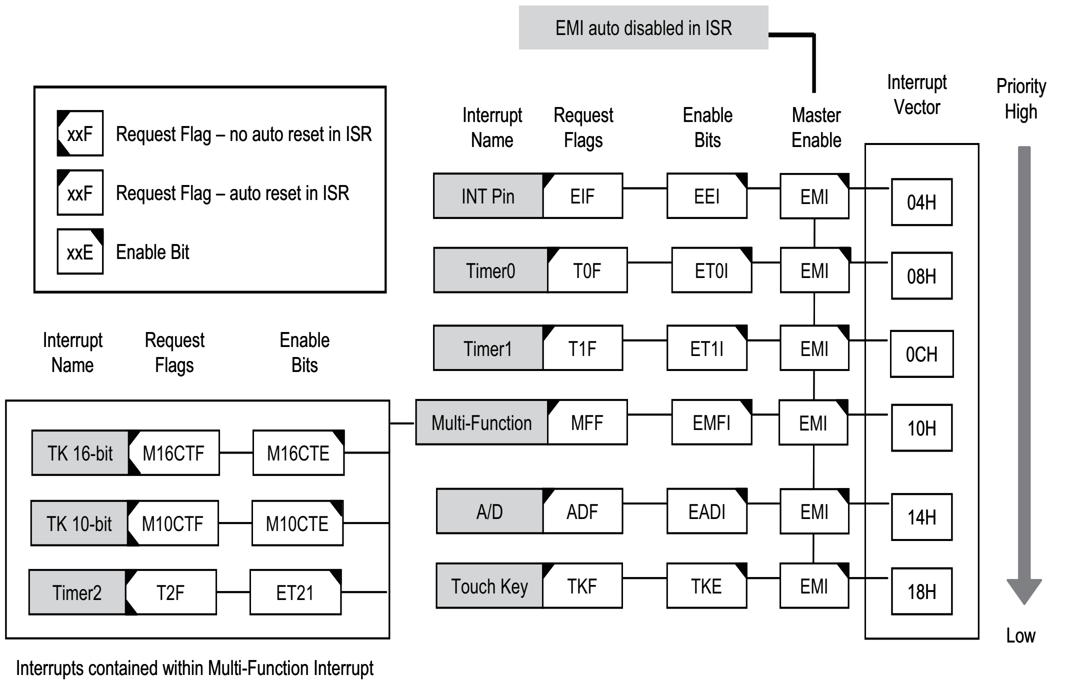
Bit 1 **M10CTE**: Touch Key Module 10-bit Counter interrupt enable
0: disable
1: enable

Bit 0 **ET2I**: Timer/Event Counter 2 interrupt enable
0: disable
1: enable

Interrupt Operation

A Timer/Event or Touch Key Counter overflow, an active edge on the external interrupt pin, A/D conversion completion, or a signal completion of the Touch Key sensor will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, 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 statement 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 instruction, 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.

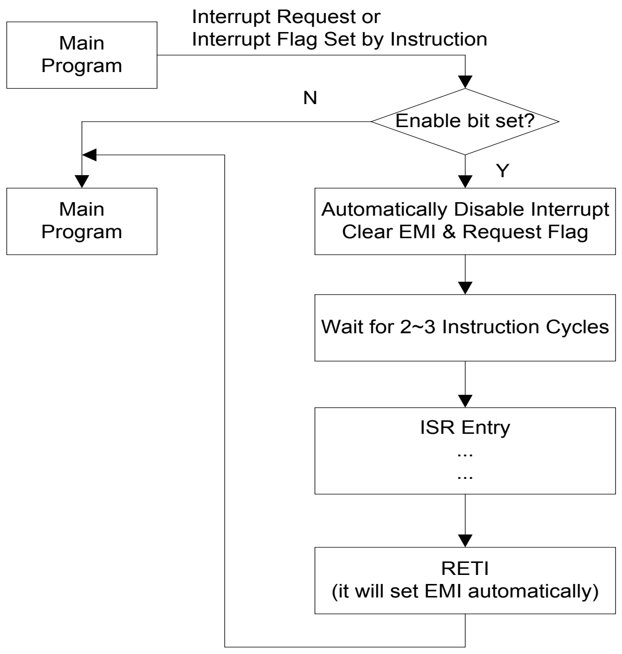
The various interrupt enable bits, together with their associated request flags, are shown in the following diagram with their order of priority.



Interrupt Structure

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the 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.

When an interrupt request is generated it takes 2 or 3 instruction cycle before the program jumps to the interrupt vector. If the device is in the Sleep or Idle Mode and is woken up by an interrupt request then it will take 3 cycles before the program jumps to the interrupt vector.


Main Program
Interrupt Request or Interrupt Flag Set by Instruction
Enable bit set?
N
Y
Main Program
Automatically Disable Interrupt Clear EMI & Request Flag
Wait for 2~3 Instruction Cycles
ISR Entry ... ...
RETI (it will set EMI automatically)

Interrupt Flow

Interrupt Priority

Interrupts, occurring in the interval between the rising edges of two consecutive T2 pulses, will be serviced on the latter of the two T2 pulses, if the corresponding interrupts are enabled. In case of simultaneous requests, the following table shows the priority that is applied. These can be masked by resetting the EMI bit.

Interrupt Source	Priority	Vector
External Interrupt 0	1	04H
Timer/Event Counter 0 Overflow	2	08H
Timer/Event Counter 1 Overflow	3	0CH
Multi Function Interrupt (Timer/Event Counter overflow 2 and Touch Key Module 10-bit/16-bit counter overflow)	4	10H
A/D Convertor Interrupt	5	14H
Touch Key Interrupt	6	18H

Interrupt Subroutine Vector

In cases where both external and internal interrupts are enabled and where an external and internal interrupt occurs simultaneously, the external interrupt will always have priority and will therefore be serviced first. Suitable masking of the individual interrupts using the interrupt registers can prevent simultaneous occurrences.

External Interrupt

For an external interrupt to occur, the global interrupt enable bit, EMI, and external interrupt enable bit, EEI, must first be set. An actual external interrupt will take place when the external interrupt request flag, EIF, is set, a situation that will occur when an edge transition appears on the external INT line. The type of transition that will trigger an external interrupt, whether high to low, low to high or both is determined by the EINTC0 and EINTC1 bits, which are bits 6 and 7 respectively, in the CTRL1 control register. These two bits can also disable the external interrupt function.

EINTC1	EINTC0	Edge Trigger Type
0	0	Disable
0	1	Falling edge trigger
1	0	Rising edge trigger
1	1	Dual edge trigger

The external interrupt pin is pin-shared with the I/O pin PB4 and can only be configured as an external interrupt pin if the corresponding external interrupt enable bit in the INTC0 register has been set and the edge trigger type has been selected using the CTRL1 register. The pin must also be setup as an input by setting the corresponding PBC.4 bit in the port control register. When the interrupt is enabled, the stack is not full and an active transition appears on the external interrupt pin, a subroutine call to the external interrupt vector at location 04H, will take place. When the interrupt is serviced, the external interrupt request flag, EIF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor connections on this pin will remain valid even if the pin is used as an external interrupt input.

Timer/Event Counter Interrupt

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, ETnI, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TnF, is set, a situation that will occur when the relevant Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter n overflow occurs, a subroutine call to the relevant timer interrupt vector, will take place. When the interrupt is serviced, the timer interrupt request flag, TnF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Multi-function Interrupt

The device has one Multi-function interrupt. Unlike the other independent interrupts, this interrupt have no independent source, but rather are formed from other existing interrupt sources, namely Touch Key 16-bit Counter Interrupt, Touch Key 10-bit Counter Interrupt and Timer/Event Counter 2 Interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flag MFF is set. The Multi-function interrupt flag will be set when any of its included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within Multi-function interrupt will occur, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function Interrupt flag will be automatically reset when the interrupt is serviced, the request flag from the original source of the Multi-function interrupt, namely Touch Key 16-bit Counter Interrupt, Touch Key 10-bit Counter Interrupt and Timer/Event Counter 2 Interrupt will not be automatically reset and must be manually reset by the application program.

A/D Converter Interrupt

The A/D Converter interrupt is initialized by setting the A/D converter request flag, caused by an end of A/D conversion. When the interrupt is enabled, the stack is not full and the ADF is set, a sub-routine call will occur. The related interrupt request flag ADF will be reset and the EMI bit cleared to disable further interrupts.

Touch Key Interrupt

The Touch Key interrupt is initialized by setting the Touch Key interrupt request flag, TKF, bit 6 of INTC1. This is caused by a signal completion of the Touch Key sensor. After the interrupt is enabled, and the stack is not full, and the TKF bit is set, a sub-routine call will occur. The related interrupt request flag TKF, will be reset and the EMI bit is cleared to disable further interrupts.

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 Power down 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 entering the Power down 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.

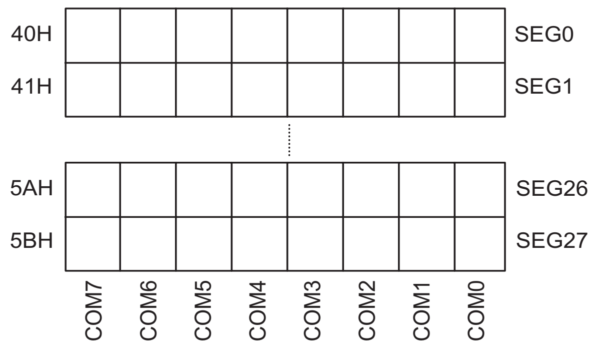
LCD Driver

For large volume applications, which incorporate an LCD in their design, the use of a custom display rather than a more expensive character based display reduces costs significantly. However, the corresponding signals required, which vary in both amplitude and time, to drive such a custom display require many special considerations for proper LCD operation to occur. The Holtek LCD Driver function, with its internal LCD signal generating circuitry and various options, will automatically generate these time and amplitude varying signals to provide a means of direct driving and easy interfacing to a range of custom LCDs.

LCD Memory

An area of Data Memory is especially reserved for use by the LCD data. This data area is known as the LCD Memory. Any data written here will be automatically read by the internal LCD driver circuits, which will in turn automatically generate the necessary LCD driving signals. Therefore any data written into the LCD Memory will be immediately reflected into the actual LCD display connected to the microcontroller. The start address of the LCD Memory is 40H; the end address of the LCD Memory is 5BH.

As the LCD Data Memory addresses overlap those of the General Purpose Data Memory, the LCD Data Memory is stored in its own memory data bank, which is different from that of the General Purpose Data Memory. The LCD Data Memory is stored in Bank 1. The Data Memory Bank is chosen by using the Bank Pointer, which is a special function register in the Data Memory, with the name, BP. When the lowest bit of the Bank Pointer has the binary value “0”, only the General Purpose Data Memory will be accessed, no read or write actions to the LCD Memory will take place. To access the LCD Memory therefore requires first that Bank 1 is selected by setting the lowest bit of the Bank Pointer to the binary value “1”. After this, the LCD Memory can then be accessed by using indirect addressing through the use of Memory Pointer MP1. With Bank 1 selected, then using MP1 to read or write to the memory area 40H~5BH, will result in operations to the LCD Memory. Directly addressing the LCD Memory is not applicable and will result in a data access to the Bank 0 General Purpose Data Memory.



LCD Memory Map

LCD Registers

A single LCD Control Registers in the Data Memory, known as LCDC, is used to control the various setup features of the LCD Driver. Various bits in this register control functions such as VA voltage, bias type, duty type as well as LCD COM/SEG selection.

LCDC Register

Bit	7	6	5	4	3	2	1	0
Name	LCDPD	VAS	—	RCS	—	CSS2	CSS1	CSS0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

Bit 7 **LCDPD**: LCD on/off control
0: LCD enable
1: LCD disable

Bit 6 **VAS**: VA voltage selection for C type LCD
0: VOREG
1: 1.5×VOREG

Bit 5 Unimplemented, read as “0”

Bit 4 **RCS**: LCD type R or C
0: R type
1: C type

Bit 3 Unimplemented, read as “0”

Bit 2~0 **CSS2~CSS0**: LCD COM[7:4]/SEG[27:24] selection, Refer to the following table for details
Note: When the charge pump output voltage is equal to VDD, the C-type LCD bias can only be selected as VA=VOREG.

When the charge pump output voltage is equal to 2×VDD, the C-type LCD bias can be selected as VA=VOREG or VA=VOREG×1.5.

When the C-type LCD voltage is selected to be pumped to VOREGx1.5, the power supply to VDD should be limited to 2.6V~5.5V.

CCS[2:0]	DUTY	COM4/SEG27	COM5/SEG26	COM6/SEG25	COM7/SEG24	SEG×COM (Maximum)
000	1/4	SEG27	SEG26	SEG25	SEG24	28×4
001	1/5	COM4	SEG26	SEG25	SEG24	27×5
010	1/6	COM4	COM5	SEG25	SEG24	26×6
011	1/7	COM4	COM5	COM6	SEG24	25×7
1××	1/8	COM4	COM5	COM6	COM7	24×8

The LCDOUT register is used to determine if the output function of LCD pins SEG0~SEG7 are used a LCD segment drivers or normal I/O operation.

LCDOUT Register

Bit	7	6	5	4	3	2	1	0
Name	LCDS7	LCDS6	LCDS5	LCDS4	LCDS3	LCDS2	LCDS1	LCDS0
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 **LCDS7**: Select SEG7 or I/O
0: I/O
1: SEG7

Bit 6 **LCDS6**: Select SEG6 or I/O
0: I/O
1: SEG6

Bit 5 **LCDS5**: Select SEG5 or I/O
0: I/O
1: SEG5

Bit 4 **LCDS4**: Select SEG4 or I/O
0: I/O
1: SEG4

Bit 3 **LCDS3**: Select SEG3 or I/O
0: I/O
1: SEG3

Bit 2 **LCDS2**: Select SEG2 or I/O
0: I/O
1: SEG2

Bit 1 **LCDS1**: Select SEG1 or I/O
0: I/O
1: SEG1

Bit 0 **LCDS0**: Select SEG0 or I/O
0: I/O
1: SEG0

LCD Clock

The LCD clock is driven by the f_{SUB} clock, which then passes through a divider, the division ratio of which is selected by the LCD clock selection bits, LCDCK1 and LCDCK0, in the CTRL0 register to provide a LCD clock frequency of $f_{SUB}/3$, f_{SUB} /4 or f_{SUB} /8. The LCD clock source f_{SUB} can be derived from the LIRC or LXT oscillator selected by the selection bit, named FSUBS. Note that the f_{SUB} clock can be enabled or disabled in the power down mode by the f_{SUB} clock control bit FSUBC in the CTRL0 register.

LCD Driver Output

The output structure of the device LCD driver can be 24×8 to 28×4. The LCD driver bias type has R and C type. The C/R type and number of COM and SEG is selected by software option. The LCD driver has a fixed 1/3 bias.

The nature of Liquid Crystal Displays require that only AC voltages can be applied to their pixels as the application of DC voltages to LCD pixels may cause permanent damage. For this reason the relative contrast of an LCD display is controlled by the actual RMS voltage applied to each pixel, which is equal to the RMS value of the voltage on the COM pin minus the voltage applied to the SEG pin. This differential RMS voltage must be greater than the LCD saturation voltage for the pixel to be on and less than the threshold voltage for the pixel to be off. The requirement to limit the DC voltage to zero and to control as many pixels as possible with a minimum number of connections, requires that both a time and amplitude signal is generated and applied to the application LCD. These time and amplitude varying signals are automatically generated by the LCD driver circuits in the microcontroller. What is known as the duty determines the number of common lines used, which are also known as backplanes or COMs. The duty, which is chosen by control bits to have a value of 1/4, 1/5, 1/6 or 1/7, 1/8 and which equates to a COM number of 3, 4, 5, 6 and 7 respectively, therefore defines the number of time divisions within each LCD signal frame. The accompanying timing diagrams depict the LCD signals generated by the microcontroller for various values of duty and bias.

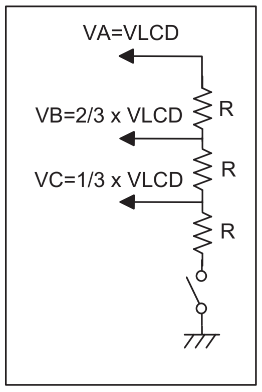
LCD Voltage Source and Biasing

The time and amplitude varying signals generated by the LCD Driver function require the generation of several voltage levels for their operation. The number of voltage levels used by the signal depends upon the value of the CSSn bits in the LCDC register. The device can have either R type or C type biasing selected via the RCS bit in the LCDC register. Selecting the C type biasing will enable an internal charge pump whose multiplying ration can be selected using an additional configuration option.

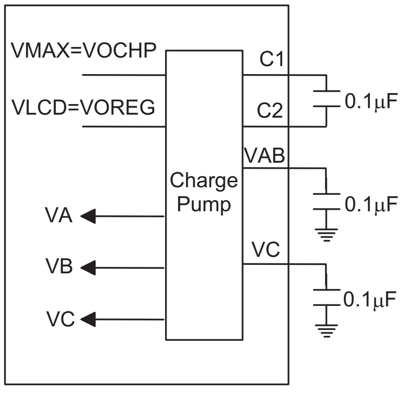
For R type biasing an external LCD voltage source is supplied by the internal VLCD biasing voltage. This could be the microcontroller power supply or some other voltage source. For the R type 1/3 bias selection, three voltage levels VSS, VA, VB and VC are utilised. The voltage VA is equal to VLCD. VB is equal to 2/3×VLCD. VC is equal to 1/3×VLCD. Note that no external capacitors or resistors are required to be connected if R type biasing is used.

For C type biasing an external LCD voltage source also is supplied by the internal VLCD biasing voltage or VMAX. The C type biasing scheme uses an internal charge pump circuit, which in the case of the 1/3 bias can generate voltages higher than what is supplied by VLCD. This feature is useful in applications where the microcontroller supply voltage is less than the supply voltage required by the LCD.

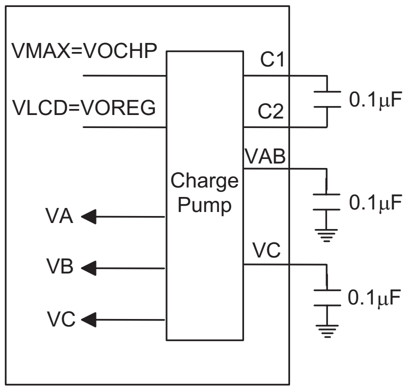
The bit "VAS" is selected for VA voltage in C type LCD. The relation of LCD voltage is as following figure.



R type and 1/3 Bias



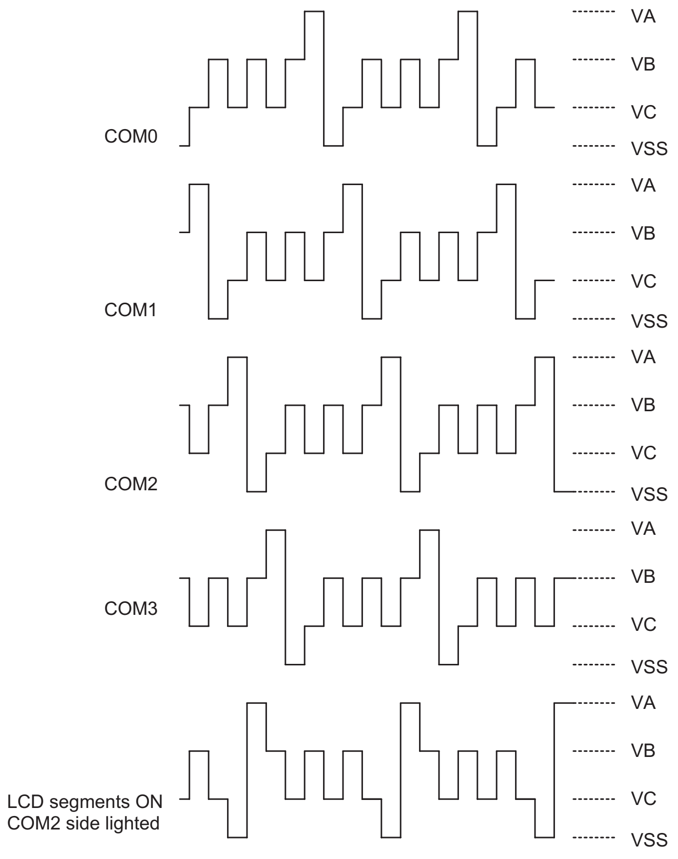
C type and 1/3 Bias,VAS="1"
VAB=3/2 x VOREG=VA
VB=VOREG
VC=1/2 x VOREG



C type and 1/3 Bias,VAS="0"
VAB=3/2 x VOREG=VB
VA=VOREG
VC=1/3 x VOREG

Note: 1. 1/3bias, VAS= 1, → VAB is VA.

2. 1/3bias, VAS= 0, → VAB is VB.



Note: 1/4 duty, 1/3 bias, R type: "VA" VLCD, "VB" 2/3 VLCD, "VC" 1/3 VLCD

LCD Drive Output (1/4 Duty)

Programming Considerations

Certain precautions must be taken when programming the LCD. One of these is to ensure that the LCD memory is properly initialised after the microcontroller is powered on. Like the General Purpose Data Memory, the contents of the LCD memory are in an unknown condition after power-on. As the contents of the LCD memory will be mapped into the actual LCD, it is important to initialise this memory area into a known condition soon after applying power to obtain a proper display pattern.

Consideration must also be given to the capacitive load of the actual LCD used in the application. As the load presented to the microcontroller by LCD pixels can be generally modeled as mainly capacitive in nature, it is important that this is not excessive, a point that is particularly true in the case of the COM lines which may be connected to many LCD pixels. The accompanying diagram depicts the equivalent circuit of the LCD.

With such a frequency chosen, the microcontroller internal LCD driver circuits will ensure that the appropriate LCD driving signals are generated to obtain a suitable LCD frame frequency.

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the OTP Program Memory device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later by the application software. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Options	
1	System Oscillator Selection - f_{SYS}: 1. HXT 2. ERC 3. HIRC
2	HIRC frequency Selection: 1. 4MHz 2. 8MHz 3. 12MHz
3	External 32kHz Oscillator Selection: 1. I/O 2. 32.768kHz external crystal
4	System Oscillator SST period Selection: 1.1024 clocks 2.2 clocks
Reset Pin Options	
5	PA7/RES Pin Options: 1. RES pin 2. I/O pin
LCD Options	
6	LCD function in Power down mode: 1. Enable 2. Disable
7	R type drive current selection: 1. 50μA 2. 100μA
Watchdog Options	
8	WDT Function 1. Always enable 2. WDT enable/disable by S/W
9	WDT clock Selection - f_S: 1. Internal 12kHz RC oscillator - LIRC 2. $f_{SYS}/4$ 3. 32.768kHz oscillator
10	CLRWDT instruction Selection: 1. instruction 2. instruction

Body Fat Measurement Function

The body fat circuit consists of a sine wave generator, an amplifier and a filter. The circuit has been designed for maximum flexibility and has a high degree of functional integration to implement a body fat measurement function.

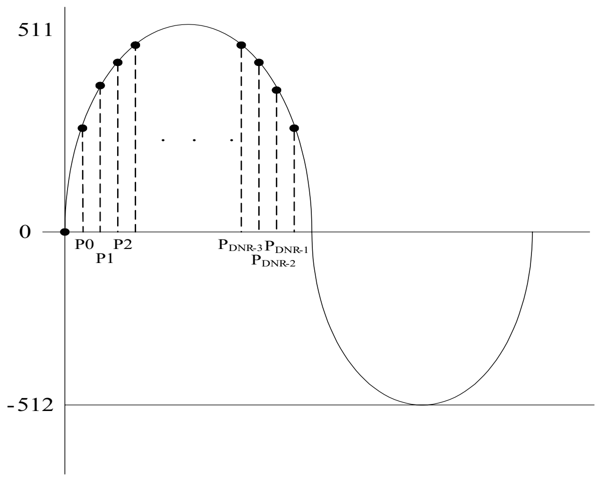
Sine Wave Generator

The sine wave generator consists of a frequency divider, counter, RAM, 10-bit DAC and OP0. The circuit can generate a sine wave output with a frequency range of 5kHz~50kHz using a 32×9 bit RAM for the sine wave pattern simulation. The frequency divider will multiply by DN/M to generate a clock for the counter. The following points must be noted to understand how the sine wave is generated:

- System clock/M = sine wave frequency
- System clock × (DN/M) = the count rate of the counter
- M must be a multiple of N and 8.
- M = N×DN
- DNR = DN/2
- DN: sine wave cycle data numerical value (DN <= 64)
- DNR: the data numerical value of the 1/2 sine wave cycle stored in RAM (DNR <= 32)

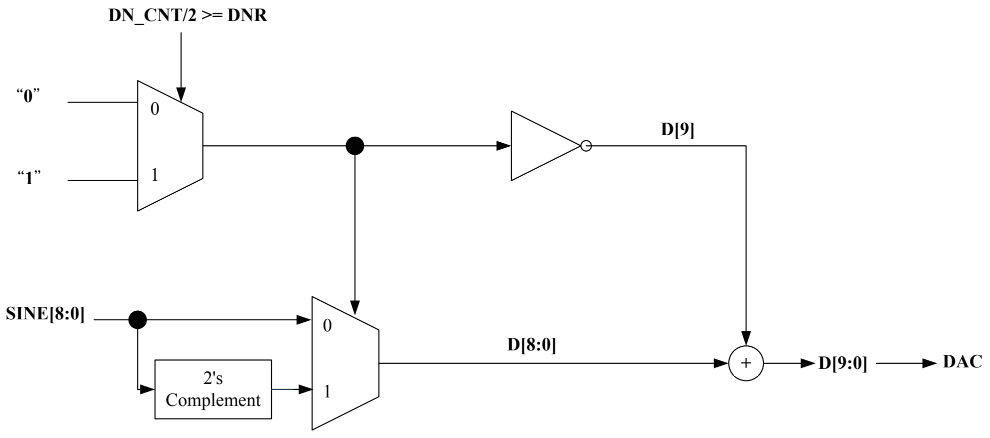
Refer to the following table and figure for more details.

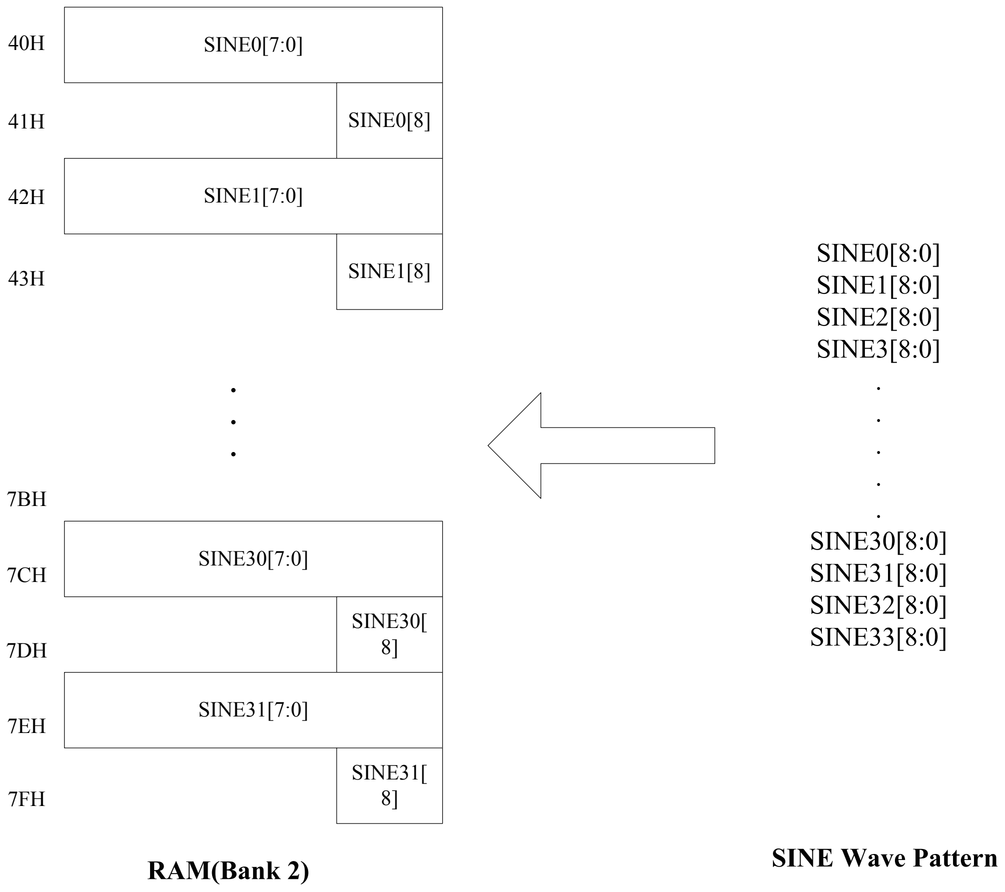
System Frequency	4MHz		8MHz		12MHz	
Sine Wave Frequency (kHz)	50	5	50	5	50	5
M	80	800	160	1600	240	2400
N	2	16	4	25	4	40
DN	40	50	40	64	60	60
DNR	20	25	20	32	30	30



Only a half sine wave pattern P0~P_{DNR-1} is generated which is stored in RAM bank 2 with an address range of 40H~7FH. The sine wave pattern data bits [7:0] are stored with even addresses while the sine wave pattern data bit [8] is stored with an odd address. Once the sine wave generator is enabled, the CPU will not be able to write or read data to/from this RAM area. The sine generator will read the RAM data and transmit it to the 10-bit DAC.

The device will read the half sine wave pattern from the RAM and generate the actual sine waveform on the SIN pin. Refer to the following diagram:





SGC Register

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

Bit 7 **SGEN**: sine generator enable bit
0: disable
1: enable
When this bit is equal to “0”, the OP0 and 10-bit DAC will be in a power down mode.

Bit 6~5 Unimplemented, read as “0”

Bit 4 **BREN**: Bias resistor enable
0: disable - power down mode
1: enable - normal mode
When this bit is enabled, it will generate a 0.5×VOREG voltage for the non-inverting input of OPA1 and OPA2.

Bit 3~0 Unimplemented, read as “0”

SGN Register

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

Bit 7~6 Unimplemented, read as “0”

Bit 5~0 **D5~D0**: system frequency multiplicator - N
Multiplicator (N) equal to D [5:0] + 1

SGDNR Register

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

Bit 7~5 Unimplemented, read as “0”

Bit 4~0 **D4~D0**: 1/2 sine wave cycle numerical value stored in RAM bank 2
DNR is equal to D [4:0] + 1

Amplifier

The amplifier consists of OP1, OP2, a 6-bit DAC and analog switches. OP2 is a differential amplifier with 1~5 multiple gain. The 6-bit DAC offers a reference voltage to the non-inverting input of OP2. The user can turn on and off switch 0 to 7 to obtain a reference resistor voltage and a body resistor voltage.

The body and reference impedance can be obtained by using the SW0 ~ SW7 switches. Refer to following table for this impedance switching.

Switch	SW0	SW1	SW2	SW3	SW4	SW5	SW6	SW7
Foot Impedance	O						O	O
Reference 1KΩ		O			O	O		
Reference 200Ω		O	O	O				

O: switch closed.

OPAC Register

Bit	7	6	5	4	3	2	1	0
Name	OPAEN	—	—	—	OP2G3	OP2G2	OP2G1	OP2G0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **OPAEN**: amplifier enable control
0: enable
1: disable
When this bit is equal to “1”, OP1, OP2 and the 6-bit DAC will be in a power down mode.

Bit 6~4 Unimplemented, read as “0”

Bit 3~0 **OP2G3~OP2G0**: OP2 gain control
0001: 1.14
0010: 1.31
0011: 1.5
0100: 1.73
0101: 2
0110: 2.33
0111: 2.75
1000: 3.285
1001: 4
1010: 5
Others: 1

SWC Register

Bit	7	6	5	4	3	2	1	0
Name	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
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 **SW7**: switch 7 control bit
0: open
1: short

Bit 6 **SW6**: switch 6 control bit
0: open
1: short

Bit 5 **SW5**: switch 5 control bit
0: open
1: short

Bit 4 **SW4**: switch 4 control bit
0: open
1: short

Bit 3 **SW3**: switch 3 control bit
0: open
1: short

Bit 2 **SW2**: switch 2 control bit
0: open
1: short

Bit 1 **SW1**: switch 1 control bit
0: open
1: short

Bit 0 **SW0**: switch 0 control bit
0: open
1: short

DACO Register

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **D5~D0**: 6-bit DAC output voltage
output voltage = $0.5V_{OREG}$ * ((D[5:0] + 1)/64)

Filter

The filter consists of CP0, a PMOS transistor and some analog switches. The filter contains a peak detection function for which an external capacitor will store the peak value for transmission to the ADC. Switches SW8 and SW9 are for capacitor discharge purposes.

FTRC Register

Bit	7	6	5	4	3	2	1	0
Name	FTREN	—	—	—	—	—	SW9	SW8
R/W	R/W	—	—	—	—	—	R/W	R/W
POR	0	—	—	—	—	—	0	0

Bit 7 **FTREN**: filter enable
0: disable
1: enable
When this bit is equal to "0", CP0 and the PMOS transistor will be in a power down mode.

Bit 6~5 Unimplemented, read as "0"

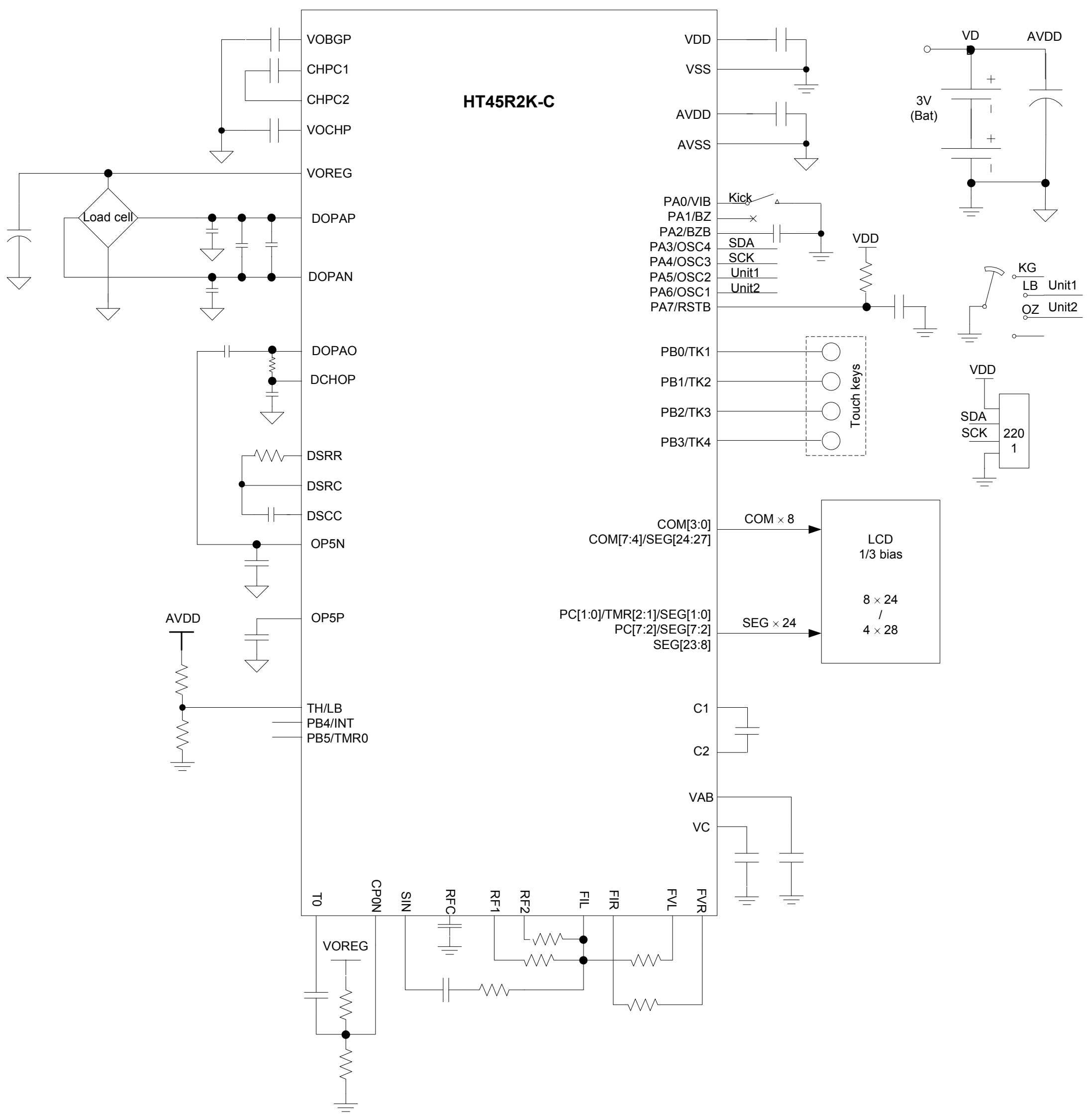
Bit 4 Reserved – must be cleared to "0"

Bit 3~2 Unimplemented, read as "0"

Bit 1 **SW9**: switch 9 control bit
0: open
1: short

Bit 0 **SW8**: switch 8 control bit
0: open
1: short

Application Circuits



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 microcontrollers, 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 Operations

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 where rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions

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

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

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRDC [m]	Read table (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
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

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

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

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

Instruction Definition

ADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C

ADCM A,[m]	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C

ADD A,[m]	Add Data Memory to ACC
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m]
Affected flag(s)	OV, Z, AC, C

ADD A,x	Add immediate data to ACC
Description	The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.
Operation	AC ← ACC + x
Affected flag(s)	OV, Z, AC, C

ADDM A,[m]	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m]
Affected flag(s)	OV, Z, AC, C

AND A,[m]	Logical AND Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "AND" [m]
Affected flag(s)	Z

AND A,x	Logical AND immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "AND"x
Affected flag(s)	Z

ANDM A,[m]	Logical AND ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.
Operation	[m] ← ACC "AND" [m]
Affected flag(s)	Z

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None

CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None

CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None

CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF

CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDTare all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF

CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDTare all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing

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.
	[m] $\leftarrow \overline{[m]}$
Operation	Z
Affected flag(s)	
	Complement Data Memory with result in ACC
CPLA [m]	Each bit of the specified Data Memory is logically complemented (1's
Description	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.
	ACC $\leftarrow \overline{[m]}$
Operation	Z
Affected flag(s)	
	Decimal-Adjust ACC for addition with result in Data Memory
DAA [m]	Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal)
Description	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.
	[m] $\leftarrow$ ACC + 00H or
Operation	[m] $\leftarrow$ ACC + 06H or
	[m] $\leftarrow$ ACC + 60H or
	[m] $\leftarrow$ ACC + 66H
	C
Affected flag(s)	
	Decrement Data Memory
DEC [m]	Data in the specified Data Memory is decremented by 1.
Description	[m] $\leftarrow$ [m] – 1
Operation	Z
Affected flag(s)	
	Decrement Data Memory with result in ACC
DECA [m]	Data in the specified Data Memory is decremented by 1. The result is stored in the
Description	Accumulator. The contents of the Data Memory remain unchanged.
	ACC $\leftarrow$ [m] – 1
Operation	Z
Affected flag(s)	

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 ← 0

Affected flag(s)	TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m]+1
Affected flag(s)	Z

INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m]+1
Affected flag(s)	Z

JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None

MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None

MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None

MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None

NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None

OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "OR" [m]
Affected flag(s)	Z

OR A,x	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "OR" x
Affected flag(s)	Z

ORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC "OR" [m]
Affected flag(s)	Z

RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None

RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None

RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None

RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i = 0~6) [m].0 ← [m].7
Affected flag(s)	None

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

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

Affected flag(s)	None

RLC [m]	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i = 0~6) [m].0 ← C C ← [m].7
Affected flag(s)	C

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

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

Affected flag(s)	C

RR [m]	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i = 0~6) [m].7 ← [m].0
Affected flag(s)	None

RRA [m]	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s)	None

RRC [m]	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i = 0~6) [m].7 ← C C ← [m].0
Affected flag(s)	C

RRCA [m]	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i $\leftarrow$ [m].(i+1); (i = 0~6) ACC.7 $\leftarrow$ C C $\leftarrow$ [m].0
Affected flag(s)	C
SBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC $\leftarrow$ ACC – [m] – $\overline{C}$
Affected flag(s)	OV, Z, AC, C
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	ACC $\leftarrow$ ACC – [m] – $\overline{C}$
Affected flag(s)	OV, Z, AC, C
SDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] $\leftarrow$ [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
Operation	instruction. ACC $\leftarrow$ [m] – 1
Affected flag(s)	Skip if ACC = 0 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].1 ← 1
Affected flag(s)	None

SIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] + 1 Skip if [m] = 0
Affected flag(s)	None

SIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] + 1 Skip if ACC = 0
Affected flag(s)	None

SNZ [m].i	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None

SUB A,[m]	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC – [m]
Affected flag(s)	OV, Z, AC, C

SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC – [m]
Affected flag(s)	OV, Z, AC, C
SUB A,x	Subtract immediate data from ACC
Description	The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC – x
Affected flag(s)	OV, Z, AC, C
SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0↔[m].7 ~ [m].4
Affected flag(s)	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3 ~ ACC.0 ← [m].7 ~ [m].4 ACC.7 ~ ACC.4 ← [m].3 ~ [m].0
Affected flag(s)	None
SZ [m]	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m] = 0
Affected flag(s)	None
SZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] Skip if [m] = 0
Affected flag(s)	None

SZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if [m].i = 0
Affected flag(s)	None

TABRDC [m]	Read table (current page) to TBLH and Data Memory
Description	The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

TABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "XOR" [m]
Affected flag(s)	Z

XORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC "XOR" [m]
Affected flag(s)	Z

XOR A,x	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "XOR" x
Affected flag(s)	Z

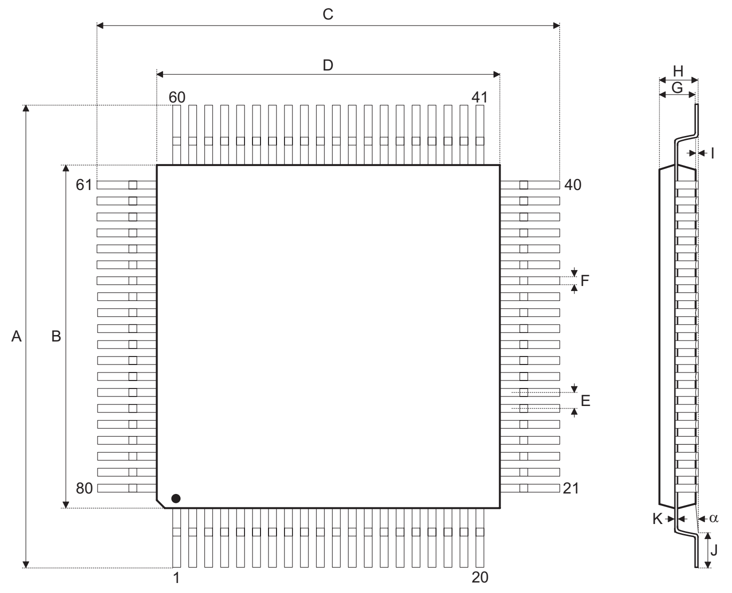
Package Information

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

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

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

80-pin LQFP (10mm×10mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.472 BSC	—
B	—	0.394 BSC	—
C	—	0.472 BSC	—
D	—	0.394 BSC	—
E	—	0.016 BSC	—
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	12 BSC	—
B	—	10 BSC	—
C	—	12 BSC	—
D	—	10 BSC	—
E	—	0.4 BSC	—
F	0.13	0.18	0.23
G	1.35	1.4	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°