HT49RA0-6

Remote Type 8-Bit MCU with LCD

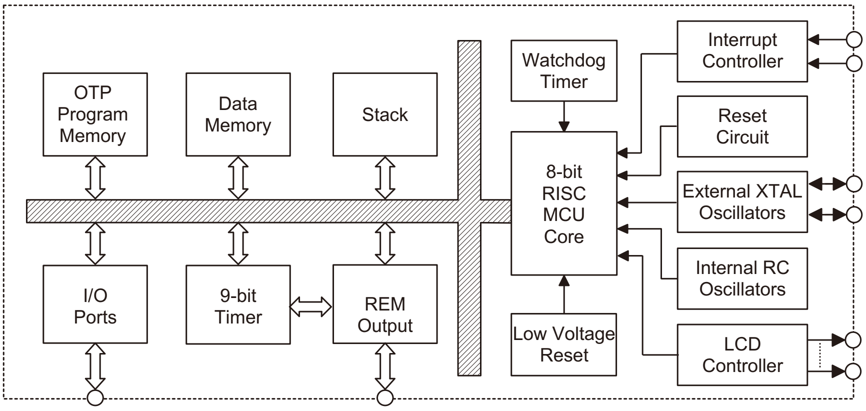
Features

- Operating voltage:
  f_{SYS} = 4MHz at V_{DD} = 2.0V~3.6V (LVR enabled)
- Oscillator types:
  - External High Frequency crystal -- HXT
  - Internal High Frequency RC -- HIRC
  - External 32768Hz crystal -- LXT
  - Internal 32kHz RC -- LIRC
- Fully integrated internal 4095kHz oscillator requires no external components
- Program Memory: 2K×16
- Data Memory: 96×8
- 4 subroutine nesting levels
- Up to 23 bidirectional I/O lines
- Two external interrupt lines shared with I/O pins
- One 8-bit programmable timer/event counter
- Real Time Clock -- RTC
- 8-bit prescaler for RTC
- One programmable carrier output -- using 9-bit timer
- Independent Carrier output pin (REM/REMDRV)
- Integrated IR driver (560mA at 3.0V)
- LCD driver with 21×2, 21×3 or 20×4 segments
- LCD display duty and bias
- Duty: 1/2, 1/3 or 1/4
- Bias: 1/2 or 1/3
- Software enable control for LCD and RTC function
- Watchdog Timer
- Low Voltage Reset/Detect function
- Power-down and wake-up features reduce power consumption
- 16-bit table read instruction
- Bit manipulation instruction
- 63 powerful instructions
- Up to 1μs instruction cycle with 4MHz system clock
- All instructions in 1 or 2 machine cycles
- 48-pin LQFP package

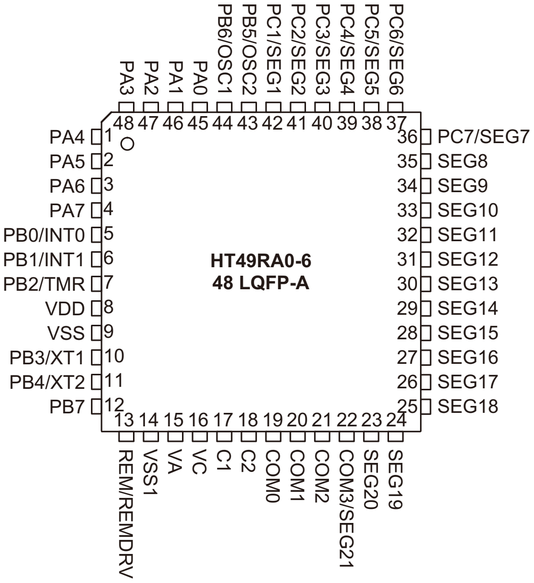
General Description

The HT49RA0-6 is 8-bit high performance, RISC architecture microcontroller devices specifically designed for multiple I/O control product applications. The advantages of low power consumption, I/O flexibility, timer functions, oscillator options, watchdog timer, power-down and wake-up functions, as well as low cost, enhance the versatility of this device to suit a wide range of application possibilities such as industrial control, consumer products, and particularly suitable for use in products such as infrared LCD remote controllers and various subsystem controllers.

Block Diagram



Pin Assignment



Pin Description

Pin Name	Function	OPT	I/T	O/T	Description
PA0~PA7	PAn	CO	ST	NMOS	General purpose I/O with pull-up resistor. Configuration option enabled wake-up.
PB0/INT0	PB0	CO	ST	NMOS	General purpose I/O with pull-up resistor. Configuration option enabled wake-up.
	INT0	CO	ST	—	External interrupt 0 input
PB1/INT1	PB1	CO	ST	NMOS	General purpose I/O with pull-up resistor. Configuration option enabled wake-up.
	INT1	CO	ST	—	External interrupt 1 input
PB2/TMR	PB2	CO	ST	NMOS	General purpose I/O with pull-up resistor. Configuration option enabled wake-up.
	TMR	TMRC	ST	—	External Timer clock input
PB3/XT1	PB3	CO	ST	NMOS	General purpose I/O with pull-up resistor. Configuration option enabled wake-up.
	XT1	CO	LXT	—	Oscillator pin
PB4/XT2	PB4	CO	ST	NMOS	General purpose I/O with pull-up resistor. Configuration option enabled wake-up.
	XT2	CO	—	LXT	Oscillator pin
PB5/OSC2	PB5	CO	ST	NMOS	General purpose I/O with pull-up resistor. Configuration option enabled wake-up.
	OSC2	CO	—	HXT	Oscillator pin
PB6/OSC1	PB6	CO	ST	NMOS	General purpose I/O with pull-up resistor. Configuration option enabled wake-up.
	OSC1	CO	HXT	—	Oscillator pin
PB7	PB7	CO	ST	NMOS	General purpose I/O without pull-up resistor. Configuration option enabled wake-up.
PC1/SEG1~ PC7/SEG7	PCn	—	ST	NMOS	General purpose I/O with pull-up resistor.
	SEGn	SEGCR	—	CMOS	LCD segment output
SEG8~SEG20	SEGn	LCDC	—	CMOS	LCD segment output
COM3/SEG21	COM3	CO	—	CMOS	LCD common output
	SEG21	CO	—	CMOS	LCD segment output
COM0~COM2	COMn	LCDC	—	CMOS	LCD common output
VA, VC, C1, C2	Vn, Cn	LCDC	—	CMOS	LCD voltage pump
REM/REMDRV	REM	TSR1	—	CMOS	Carrier output
	REMDRV	TSR1	—	NMOS	High sink carrier output
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground
VSS1	VSS	—	PWR	—	Ground, of REM/REMDRV

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

Absolute Maximum Ratings

Supply Voltage	V_{SS}–0.3V to V_{SS}+4.0V
Input Voltage	V_{SS}–0.3V to V_{DD}+0.3V
I_{OL} Total	150mA
Total Power Dissipation	500mW
Storage Temperature	–50°C to 125°C
Operating Temperature	–20°C to 70°C
I_{OH} Total	–100mA

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

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	LVR enable	2.0	—	3.6	V
I_{DD}	Operating Current (RC OSC)	3V	No load, f_{SYS}=4MHz	—	0.7	1.5	mA
I_{STB1}	Standby Current (*f_S=T1=f_{SYS})	3V	No load, system HALT, LCD off at HALT	—	0.1	1.0	μA
I_{STB2}	Standby Current (*f_S=32.768kHz,QOSC=1)	3V	No load, system HALT, LCD On at HALT, C type	—	2.5	5.0	μA
I_{STB3}	Standby Current (*f_S=WDT RC OSC)	3V	No load, system HALT LCD On at HALT, C type	—	2.0	5.0	μA
V_{IL1}	Input Low Voltage for I/O Ports, TMR, INT0 and INT1	3V	—	0	—	0.2V_{DD}	V
V_{IH1}	Input High Voltage for I/O Ports, TMR, INT0 and INT1	3V	—	0.8V_{DD}	—	V_{DD}	V
I_{OL1}	PA0~PA7, PB0~PB6, PC1~PC7, REM sink current	3V	V_{OL}=0.1V_{DD}	6	12	—	mA
I_{OH1}	REM Source Current	3V	V_{OH}=0.9V_{DD}	–5	–7	—	mA
I_{OL2}	LCD Common and Segment Current	3V	V_{OL}=0.1V_{DD}	210	420	—	μA
I_{OH2}	LCD Common and Segment Current	3V	V_{OH}=0.9V_{DD}	–80	–160	—	μA
I_{OL3}	PB7 Sink Current	3V	V_{OL}=0.1V_{DD}	0.8	1.2	—	mA
I_{OL4}	REMDRV Sink Current	3V	V_{OL}=0.6V, Ta=25°C	500	560	—	mA
R_{PH}	Pull-high Resistance of I/O Ports	3V	—	100	150	200	kΩ
V_{LVR}	Low Voltage Reset Voltage	—	Ta=25°C	1.8	1.9	2.0	V
V_{LVD}	Low Voltage Detector Voltage	—	Ta=25°C	2.0	2.1	2.2	V

Note: ″*″ for the value of VA refer to the LCD driver section.

″*f_S″ please refer to WDT clock option

A.C. Characteristics

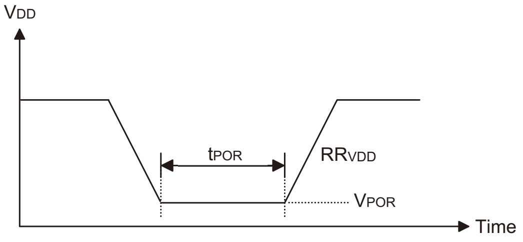
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock	1.8V~ 3.6V	Ta= -20°C~70°C	400	—	4000	kHz
f_{HIRC}	System Clock (HIRC)	2.2V~ 3.6V	Ta= -10°C~50°C	-1%	4095	+1%	kHz
f_{LXT}	LXT oscillator frequency	—	—	—	32768	—	Hz
f_{LIRC}	Cystem Clock (LIRC)	3V	Ta=25°C	-10%	32	+10%	kHz
f_{TIMER}	Timer I/P frequency (TMR)	3V	—	0	—	4000	kHz
t_{SST}	System Start-up Timer Period	—	Power-up or Wake-up from HALT (HXT)	—	128	—	$^*t_{SYS}$
			Power-up from HALT (HIRC)	—	128	—	$^*t_{SYS}$
			Wake-up from HALT (HIRC)	—	2	—	$^*t_{SYS}$
t_{INT}	Interrupt pulse width	—	—	1	—	—	μs
t_{LVR}	Low voltage width to reset	—	—	0.25	1.00	2.00	ms
f_{READYB}	REMDRV output function stable time; polling READYB register bit=0	—	Wake-up from HALT or change REMDRV register bit from 1 to 0	—	250	—	μs

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

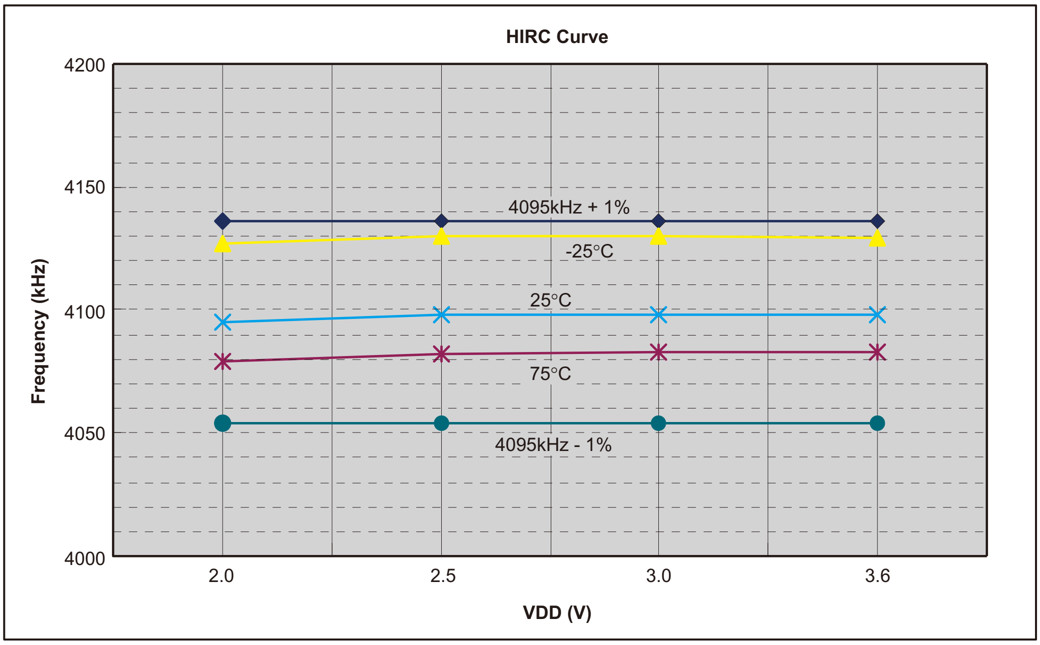
Power-on Reset Characteristics

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 raising rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for VDD Stays at V_{POR} to Ensure Power-on Reset	—	—	1	—	—	ms

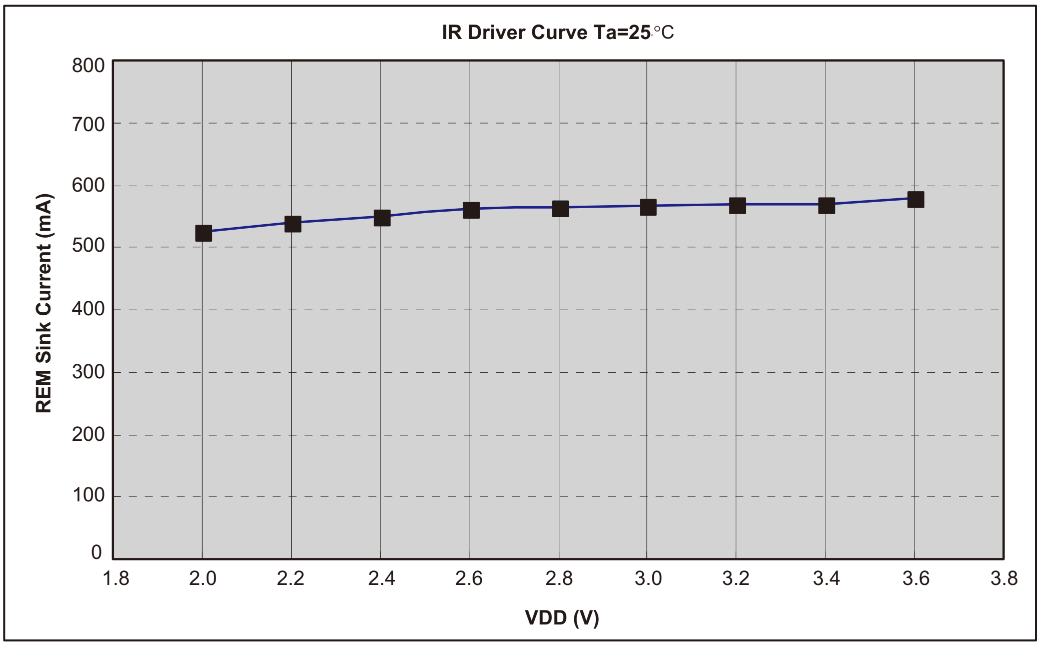


Characteristics Curves

HIRC Oscillator Voltage/Temperature vs. Frequency



IR Sink Current vs. VDD



Functional Description

Execution Flow

The main system clock is derived from either an external crystal oscillator which requires the connection of the external crystal or resonator or an internal RC oscillator which requires no external component for its operation. It is internally divided into four non-overlapping clocks. One instruction cycle consists of four system clock cycles.

Instruction fetching and execution are pipelined in such a way that a fetch takes one instruction cycle while decoding and execution takes the next instruction cycle. However, the pipelining scheme causes each instruction to effectively execute within one cycle. If an instruction changes the program counter, two cycles are required to complete the instruction.

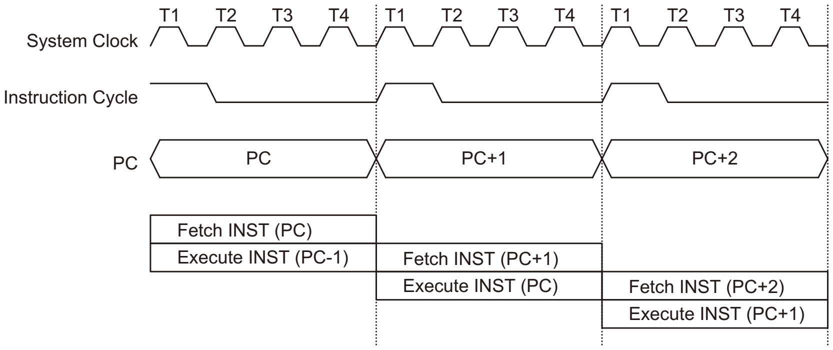
Program Counter – PC

The program counter (PC) is 11 bits wide and controls the sequence in which the instructions stored in the program ROM are executed. The contents of the PC can specify a maximum of 2048 addresses.

After accessing a program memory word to fetch an instruction code, the value of the PC is incremented by one. The PC then points to the memory word containing the next instruction code.

When executing a jump instruction, conditional skip execution, loading a PCL register, a subroutine call, an initial reset, an internal interrupt, an external interrupt, or returning from a subroutine, the PC manipulates the program transfer by loading the address corresponding to each instruction.

The conditional skip is activated by instructions. Once the condition is met, the next instruction, fetched during the current instruction execution, is discarded and a dummy cycle replaces it to get a proper instruction; otherwise proceed with the next instruction.



Execution Flow

Mode	Program Counter										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0
External Interrupt 0	0	0	0	0	0	0	0	0	1	0	0
External Interrupt 1	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter overflow	0	0	0	0	0	0	0	1	1	0	0
Time Base Interrupt	0	0	0	0	0	0	1	0	0	0	0
RTC Interrupt	0	0	0	0	0	0	1	0	1	0	0
Skip	Program Counter + 2										
Loading PCL	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return From Subroutine	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Note: *10~*0: Program counter bits
#10~#0: Instruction code bits
S10~S0: Stack register bits
@7~@0: PCL bits

The lower byte of the PC (PCL) is a readable and writeable register (06H). Moving data into the PCL performs a short jump. The destination is within 256 locations. When a control transfer takes place, an additional dummy cycle is required.

Program Memory – ROM

The program memory (ROM) is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 2048 × 16 bits which are addressed by the program counter and table pointer.

Certain locations in the ROM are reserved for special usage:

- Location 000H

  Location 000H is reserved for program initialization. After chip reset, the program always begins execution at this location.

- Location 004H

  Location 004H is reserved for the external interrupt service program. If the INT0 input pin is activated, and the interrupt is enabled, and the stack is not full, the program begins execution at location 004H.

- Location 008H

  Location 008H is reserved for the external interrupt service program also. If the INT1 input pin is activated, and the interrupt is enabled, and the stack is not full, the program begins execution at location 008H.

- Location 00CH

  Location 00CH is reserved for the Timer/Event Counter interrupt service program. If a timer interrupt results from a Timer/Event Counter overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.

- Location 010H

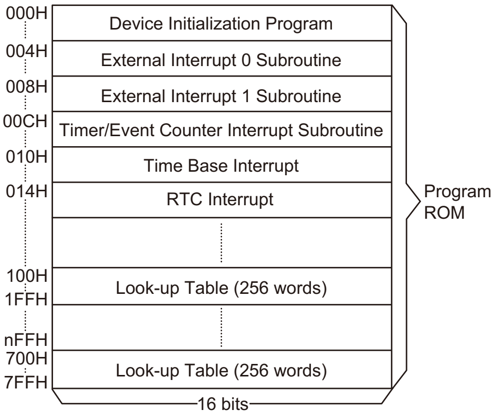
  Location 010H is reserved for the Time Base interrupt service program. If a Time Base interrupt occurs, and the interrupt is enabled, and the stack is not full, the program begins execution at location 010H.

- Location 014H

  Location 014H is reserved for the real time clock interrupt service program. If a real time clock interrupt occurs, and the interrupt is enabled, and the stack is not full, the program begins execution at location 014H.

- Table location

  Any location in the ROM can be used as a look-up table. The instructions TABRDC [m] (the current page, 1 page=256 words) and TABRDL [m] (the last page) transfer the contents of the lower-order byte to the specified data memory, and the contents of the higher-order byte to TBLH (Table Higher-order byte register) (08H). Only the destination of the lower-order byte in the table is well-defined; the other bits of the table word are all transferred to the lower portion of TBLH, and the remaining 2 bit is read as "0". The TBLH is read only, and the table pointer (TBLP) is a read/write register (07H), indicating the table location. Before accessing the table, the location should be placed in TBLP. All the table related instructions require 2 cycles to complete the operation. These areas may function as a normal ROM depending upon the user's requirements.


000H
Device Initialization Program
004H
External Interrupt 0 Subroutine
008H
External Interrupt 1 Subroutine
00CH
Timer/Event Counter Interrupt Subroutine
010H
Time Base Interrupt
014H
RTC Interrupt
100H
1FFH
Look-up Table (256 words)
nFFH
700H
7FFH
Look-up Table (256 words)
Program ROM
16 bits

Note: n ranges from 0 to 6

Program Memory

<table>
<tr><th rowspan="2">Instruction(s)</th><th colspan="11">Table Location</th></tr>
<tr><th>*10</th><th>*9</th><th>*8</th><th>*7</th><th>*6</th><th>*5</th><th>*4</th><th>*3</th><th>*2</th><th>*1</th><th>*0</th></tr>
<tr><td>TABRDC [m]</td><td>P10</td><td>P9</td><td>P8</td><td>@7</td><td>@6</td><td>@5</td><td>@4</td><td>@3</td><td>@2</td><td>@1</td><td>@0</td></tr>
<tr><td>TABRDL [m]</td><td>1</td><td>1</td><td>1</td><td>@7</td><td>@6</td><td>@5</td><td>@4</td><td>@3</td><td>@2</td><td>@1</td><td>@0</td></tr>
</table>

Table Location

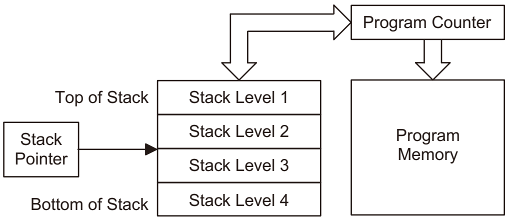
Note: *10~*0: Table location bits
@7~@0: Table pointer bits

P10~P8: Current program Counter bits

Stack Register – STACK

The stack register is a special part of the memory used to save the contents of the program counter. The stack is organized into 4 levels and is neither part of the data nor part of the program, and is neither readable nor writeable. Its activated level is indexed by a stack pointer (SP) and is neither readable nor writeable. At a commencement of a subroutine call or an interrupt acknowledgment, the contents of the program counter is pushed onto the stack. At the end of the subroutine or interrupt routine, signaled by a return instruction (RET or RETI), the contents of the program counter is restored to its previous value from the stack. After chip reset, the SP will point to the top of the stack.

If the stack is full and a non-masked interrupt takes place, the interrupt request flag is recorded but the acknowledgment is still inhibited. Once the SP is decremented (by RET or RETI), the interrupt is serviced. This feature prevents stack overflow, allowing the programmer to use the structure easily. Likewise, if the stack is full, and a ″CALL″ is subsequently executed, a stack overflow occurs and the first entry is lost (only the most recent 4 return addresses are stored).



Data Memory – RAM

The data memory is divided into two functional groups: special function registers and general purpose data memory (96×8). Most of them are read/write, but some are read only. The unused space before 20H is reserved for future expanded usage and reading these locations will return the result 00H. The general purpose data memory, addressed from 20H to 7FH, is used for data and control information under instruction command. The areas in the RAM can directly handle arithmetic, logic, increment, decrement, and rotate operations. Except some dedicated bits, each bit in the RAM can be set and reset by ″SET [m].i″ and ″CLR [m].i″. They are also indirectly accessible through the Memory pointer register 0 (MP0; 01H) or the Memory pointer register 1 (MP1; 03H).

	Bank 0	Bank 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1	
04H	BP	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	RTCC	
0AH	STATUS	
0BH	INTC0	
0CH	CTRL	
0DH	TMR	
0EH	TMRC	
0FH	WDTC	
10H	▨	
11H	▨	
12H	PA	
13H	▨	
14H	PB	
15H	▨	
16H	PC	
17H	SEGCR	
18H	TSR0	
19H	TSR1	
1AH	CARL0	
1BH	CARL1	
1CH	CARH0	
1DH	CARH1	
1EH	INTC1	
1FH	LCDC	
20H ⋮ 3FH	General Purpose Data Memory (96 Bytes)	▨
40H ⋮ 55H		LCD RAM
56H ⋮ 7FH		▨

▨ :unimplemented, read as "0"

RAM Mapping

Indirect Addressing Register – IAR0, IAR1

Location 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation of [00H] and [02H] accesses the RAM pointed to by MP0 (01H) and MP1 (03H) respectively. Reading location 00H or 02H indirectly returns the result 00H. While, writing it indirectly leads to no operation. The function of data movement between two indirect addressing registers is not supported. The memory pointer registers, MP0 and MP1, are both 7-bit registers used to access the RAM by combining corresponding indirect addressing registers. MP0 can only be applied to data memory, while MP1 can be applied to data memory and LCD display memory.

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank 0, while MP1 and IAR1 are used to access data from all banks according to BP register. Direct Addressing can only be used with Bank 0, all other Banks must be addressed indirectly using MP1 and IAR1.

Bank Pointer – BP

Depending upon which device is used, 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 0 of the Bank Pointer is used to select Data Memory Banks 0~1. The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the Power-down Mode, in which case, the Data Memory bank remains unaffected. It should be noted that the Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within any bank. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from banks other than Bank 0 must be implemented using Indirect addressing mode. As both the Program Memory and Data Memory share the same Bank Pointer Register, care must be taken during programming.

Accumulator – ACC

The accumulator (ACC) is related to the ALU operations. It is also mapped to location 05H of the RAM and is capable of operating with immediate data. The data movement between two data memory locations must pass through the ACC.

Arithmetic and Logic Unit – ALU

This circuit performs 8-bit arithmetic and logic operations and provides the following functions:

- Arithmetic operations (ADD, ADC, SUB, SBC, DAA)
- Logic operations (AND, OR, XOR, CPL)
- Rotation (RL, RR, RLC, RRC)
- Increment and Decrement (INC, DEC)
- Branch decision (SZ, SNZ, SIZ, SDZ etc.)

The ALU not only saves the results of a data operation but also changes the status register.

Status Register – STATUS

The status register (0AH) is of 8 bits wide and contains, a carry flag (C), an auxiliary carry flag (AC), a zero flag (Z), an overflow flag (OV), a power down flag (PDF), and a watchdog time-out flag (TO). It also records the status information and controls the operation sequence.

Except the TO and PDF flags, bits in the status register can be altered by instructions similar to other registers. Data written into the status register does not alter the TO or PDF flags. Operations related to the status register, however, may yield different results from those intended. The TO and PDF flags can only be changed by a Watchdog Timer overflow, chip power-up, or clearing the Watchdog Timer and executing the ″HALT″ instruction. The Z, OV, AC, and C flags reflect the status of the latest operations.

Bit No.	Label	Function
0	BP0	BP0: Select Data Memory Banks 0: Bank 0 1: Bank 1
1~7	—	Unused bit, read as ″0″

BP Register

Bit No.	Label	Function
0	C	C is set if the operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
1	AC	AC is set if the operation results in a carry out of the low nibbles in addition or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
2	Z	Z is set if the result of an arithmetic or logic operation is zero; otherwise Z is cleared.
3	OV	OV is set if the operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
4	PDF	PDF is cleared by either a system power-up or executing the ″CLR WDT″ instruction. PDF is set by executing the ″HALT″ instruction.
5	TO	TO is cleared by a system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.
6, 7	—	Unused bit, read as ″0″

Status (0AH) Register

On entering the interrupt sequence or executing the subroutine call, the status register will not be automatically pushed onto the stack. If the contents of the status is important, and if the subroutine is likely to corrupt the status register, the programmer should take precautions and save it properly.

Interrupts

The devices provides two external interrupts, one internal timer/event counter interrupts, an internal time base interrupt, and an internal real time clock interrupt. The interrupt control register 0 (INTC0;0BH) and interrupt control register 1 (INTC1;1EH) both contain the interrupt control bits that are used to set the enable/disable status and interrupt request flags.

Once an interrupt subroutine is serviced, other interrupts are all blocked (by clearing the EMI bit). This scheme may prevent any further interrupt nesting. Other interrupt requests may take place during this interval, but only the interrupt request flag will be recorded. If a certain interrupt requires servicing within the service routine, the EMI bit and the corresponding bit of the INTC0 or of INTC1 may be set in order to allow interrupt nesting. Once 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 should be prevented from becoming full.

All these interrupts can support a wake-up function. As an interrupt is serviced, a control transfer occurs by pushing the contents of the program counter onto the stack followed by a branch to a subroutine at the specified location in the ROM. Only the contents of the program counter is pushed onto the stack. If the contents of the register or of the status register (STATUS) is altered by the interrupt service program which corrupts the desired control sequence, the contents should be saved in advance.

External interrupts are triggered by a high to low or low to high or both transition of INT0 or INT1, and the related interrupt request flag (EIF0;bit 4 of INTC0, EIF1;bit 5 of INTC0) is set as well. After the interrupt is enabled, the stack is not full, and the external interrupt is active, a subroutine call to location 04H or 08H occurs. The interrupt request flag (EIF0 or EIF1) and EMI bits are all cleared to disable other interrupts.

The internal Timer/Event Counter interrupt is initialized by setting the Timer/Event Counter interrupt request flag (TF;bit 6 of INTC0), which is normally caused by a timer overflow. After the interrupt is enabled, and the stack is not full, and the TF bit is set, a subroutine call to location 0CH occurs. The related interrupt request flag (TF) is reset, and the EMI bit is cleared to disable further interrupts.

The time base interrupt is initialized by setting the time base interrupt request flag (TBF;bit 4 of INTC1), that is caused by a regular time base signal. After the interrupt

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1=enabled; 0=disabled)
1	EEI0	Controls the external interrupt 0 (1=enabled; 0=disabled)
2	EEI1	Controls the external interrupt 1 (1=enabled; 0=disabled)
3	ETI	Controls the Timer/Event Counter interrupt (1=enabled; 0=disabled)
4	EIF0	External interrupt 0 request flag (1=active; 0=inactive)
5	EIF1	External interrupt 1 request flag (1=active; 0=inactive)
6	TF	Internal Timer/Event Counter request flag (1=active; 0=inactive)
7	—	Unused bit, read as "0"

INTC0 (0BH) Register

Bit No.	Label	Function
0	ETBI	Controls the time base interrupt (1=enabled; 0:disabled)
1	ERTI	Controls the real time clock interrupt (1=enabled; 0:disabled)
2, 3	—	Unused bit, read as "0"
4	TBF	Time base request flag (1=active; 0=inactive)
5	RTF	Real time clock request flag (1=active; 0=inactive)
6, 7	—	Unused bit, read as "0"

INTC1 (1EH) Register

is enabled, and the stack is not full, and the TBF bit is set, a subroutine call to location 10H occurs. The related interrupt request flag (TBF) is reset and the EMI bit is cleared to disable further interrupts.

The real time clock interrupt is initialized by setting the real time clock interrupt request flag (RTF; bit 5 of INTC1), that is caused by a regular real time clock signal. After the interrupt is enabled, and the stack is not full, and the RTF bit is set, a subroutine call to location 14H occurs. The related interrupt request flag (RTF) is reset and the EMI bit is cleared to disable further interrupts.

During the execution of an interrupt subroutine, other interrupt acknowledgments are all held until the ″RETI″ instruction is executed or the EMI bit and the related interrupt control bit are set both to 1 (if the stack is not full). To return from the interrupt subroutine, ″RET″ or ″RETI″ may be invoked. RETI sets the EMI bit and enables an interrupt service, but RET does not.

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

Interrupt Source	Priority	Vector
External interrupt 0	1	04H
External interrupt 1	2	08H
Timer/Event Counter overflow	3	0CH
Time base interrupt	4	10H
Real time clock interrupt	5	14H

The EMI, EEI0, EEI1, ETI, ETBI and ERTI are all used to control the enable/disable status of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags (RTF, TBF, TF, EIF1, EIF0) are all set, they remain in the INTC1 or INTC0 respectively until the interrupts are serviced or cleared by a software instruction.

It is recommended that a program not use the ″CALL subroutine″ within the interrupt subroutine. It′s because interrupts often occur in an unpredictable manner or require to be serviced immediately in some applications. At this time, if only one stack is left, and enabling the interrupt is not well controlled, operation of the ″call″ in the interrupt subroutine may damage the original control sequence.

Oscillator Configuration

In this device there are two methods of generating the system clock, one external crystal oscillator and one internal RC oscillator.

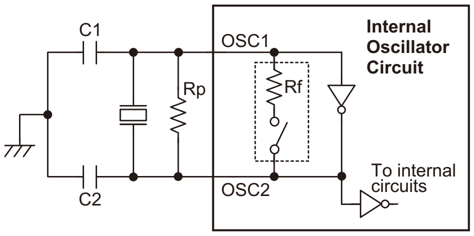
In addition to the main system clock oscillators, there is an external 32768Hz crystal (LXT) oscillator to provide the clock sources for Real Time and Time Base Interrupts and LCD display.

- External Crystal/Resonator Oscillator – HXT

  The External Crystal/Ceramic System Oscillator is one of the system oscillator choices, which is selected via a configuration option. For the crystal oscillator configuration, the connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation. Two external capacitors may be required to be connected as shown. However, the feedback resistor named Rf shown in the following diagram for the crystal oscillator to oscillate properly can be selected as either an internally or externally connected type via a configuration option. When the external connection type of the feedback resistor is selected, the recommended value of the external connected feedback resistor ranges from 300kΩ to 500kΩ. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer′s specification.

Crystal Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
4MHz	8pF	10pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values



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

Crystal/Resonator Oscillator – HXT

- Internal RC Oscillator – HIRC

  The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 4095kHz. 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 ranging from 1.8V to 3.6V and in a temperature range from -20°C to 50°C degrees, the fixed oscillation frequency of 4095kHz 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 PB5 and PB6 are free for use as normal I/O pins.

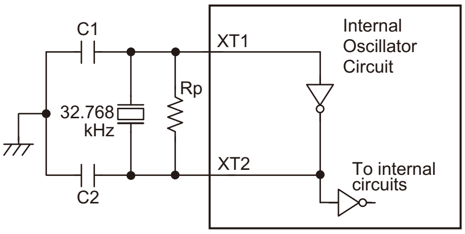
- External 32.768kHz Crystal Oscillator – LXT

  The External 32.768kHz Crystal Oscillator is one of the oscillator choices for WDT function, Real Time and Time Base interrupts and LCD display, which is selected via configuration options. This clock source has a fixed frequency of 32.768 kHz and requires a 32.768 kHz 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.

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

  Some configuration options determine if the XT1/XT2 pins are used for the LXT oscillator or as I/O pins.

  - If the LXT oscillator is not used for any clock source, the XT1/XT2 pins should be left as floating states.
  - If the LXT oscillator is used for any clock source, the 32.768 kHz crystal should be connected to the XT1/XT2 pins.



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

External Oscillator – LXT

LXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
32.768kHz	10pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. R_P=5M~10MΩ is recommended.		

32.768kHz 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 QOSC bit in the RTCC register.

QOSC Bit	LXT Mode
0	Quick Start
1	Low-power

After power-on the QOSC 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 QOSC 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 QOSC bit high about 2 seconds after power-on.

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

Internal Low Speed Oscillator – LIRC

The LIRC is a fully self-contained on-chip RC oscillator with a typical frequency of 32kHz at 3V requiring no external components. 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. When the device enters the Power-Down Mode, the system clock will stop running but the LIRC oscillator continues to run and to keep the watchdog active if the WDT function is enabled. However, to preserve power in certain applications the LIRC can be disabled via a configuration option together with software register control bits.

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 Watchdog Timer clock source is provided by the internal clock which is supplied by the LIRC oscillator. The LIRC internal oscillator has an approximate period of 32kHz at a supply voltage of 5V. However, it should be noted that this specified internal clock period can vary with VDD, temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

- Watchdog Timer control register

  A single register, WDTC, controls the required timeout period as well as the enable/disable operation. This register together with the corresponding configuration option control the overall operation of the Watchdog Timer.

- WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT function software control
10101: Disabled
01010: Enabled
other: Reset MCU
When these bits are changed by the environmental noise to reset the microcontroller, the reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the CTRL register will be set to 1.

Bit 2~0 **WS2~WS0**: WDT Time-out period selection
000: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$

- CTRL Register

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

Bit 7~1 Unimplemented, read as 0

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

- Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. Some of the Watchdog Timer options, such as always on select and clear instruction type are selected using configuration options. With regard to the Watchdog Timer enable/disable function, there are also five bits, WE4~WE0, in the WDTC register to offer additional enable/disable and reset control of the Watchdog Timer. If the WDT configuration option is determined that the WDT function is always enabled, the WE4~WE0 bits still have effects on the WDT function. When the WE4~WE0 bits value is equal to 01010B or 10101B, the WDT function is enabled. However, if the WE4~WE0 bits are changed to any other values except 01010B and 10101B, which is caused by the environmental noise, it will reset the microcontroller after 2~3 LIRC clock cycles. If the WDT configuration option is determined that the WDT function is controlled by the WDT control register, the WE4~WE0 values can determine which mode the WDT operates in. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B. The WDT function will be enabled if the WE4~WE0 bits value is equal to 01010B. If the WE4~WE0 bits are set to any other values by the environmental noise, except 01010B and 10101B, it will reset the device after 2~3 LIRC clock cycles. After power on these bits will have the value of 01010B.

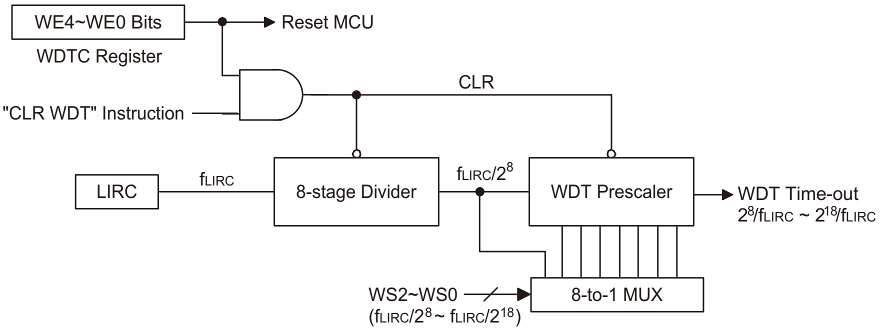
WDT Configuration Option	WE4 ~ WE0 Bits	WDT Function
Always Enable	01010B or 10101B	Enable
	Any other value	Reset MCU
Controlled by WDT Control Register	10101B	Disable
	01010B	Enable
	Any other value	Reset MCU

Watchdog Timer Enable/Disable Control

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

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

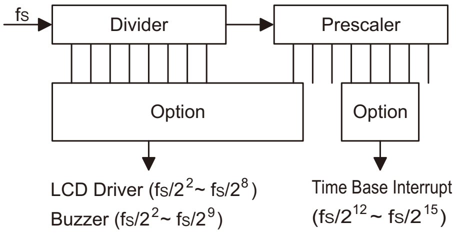
The maximum time out period is when the 2^{18} division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 second for the 2^{18} division ratio, and a minimum timeout of 7.8ms for the 2^{8} division ration.



Watchdog Timer

Time Base

The time base offers a periodic time-out period to generate a regular internal interrupt. Its time-out period ranges from $f_S/2^{12}$ to $f_S/2^{15}$ selected by options. If time base time-out occurs, the related interrupt request flag (TBF; bit 4 of INTC1) is set. But if the interrupt is enabled, and the stack is not full, a subroutine call to location 10H occurs.



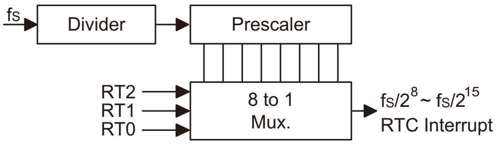
Time Base

Real Time Clock – RTC

The real time clock (RTC) is operated in the same manner as the time base that is used to supply a regular internal interrupt. Its time-out period ranges from $f_S/2^8$ to $f_S/2^{15}$ by software programming . Writing data to RT2, RT1 and RT0 (bit2, 1, 0 of RTCC;09H) yields various time-out periods. If the RTC time-out occurs, the related interrupt request flag (RTF; bit 5 of INTC1) is set. But if the interrupt is enabled, and the stack is not full, a subroutine call to location 14H occurs. The real time clock time-out signal also can be applied to be a clock source of Timer/Event Counter for getting a longer time-out period.

RT2	RT1	RT0	RTC Clock Divided Factor
0	0	0	2^{8*}
0	0	1	2^{9*}
0	1	0	2^{10*}
0	1	1	2^{11*}
1	0	0	2^{12}
1	0	1	2^{13}
1	1	0	2^{14}
1	1	1	2^{15}

Note: ″*″ not recommended to be used



Real Time Clock

Power Down Operation – HALT

The HALT mode is initialized by the ″HALT″ instruction and results in the following.

- The system oscillator turns off but the LIRC OSC keeps running (if WDT is enabled or the LIRC oscillator or real time clock is selected).
- The contents of the on-chip RAM and of the registers remain unchanged.
- The WDT is cleared and start recounting (if WDT is enabled).
- All I/O ports maintain their original status.
- The PDF flag is set but the TO flag is cleared.

The system quits the HALT mode by an interrupt, an external falling edge signal on port B, or a WDT overflow. The WDT overflow performs a ″warm reset″. After examining the TO and PDF flags, the reason for chip 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 Stack Pointer, and leaves the others at their original state.

The port A and port B wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A and port B can be independently selected to wake-up the device by option. Awakening from an I/O port stimulus, the program resumes execution of the next instruction. On the other hand, 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 resumes 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 awaken using that interrupt.

If wake-up events occur, it takes a certain period known as the t_{SST} period 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 minimize power consumption, all the I/O pins should be carefully managed before entering the HALT status.

When at HALT state and f_S=f_{SYS}/4, the LCD and RTC will be turned off no matter the bit value of (LCDEN, RTCEN).

Reset

There are three ways in which reset may occur.

- Power-on reset
- LVR is reset during normal operation
- WDT time-out is reset during normal operation

The WDT time-out during HALT differs from other chip reset conditions, for it can perform a ″warm reset″ that resets only the program counter and stack pointer and leaves the other circuits at their original state. Some registers remain unaffected during any other reset conditions. Most registers are reset to the ″initial condition″ once the reset conditions are met. Examining the PDF and TO flags, the program can distinguish between different ″chip resets″.

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

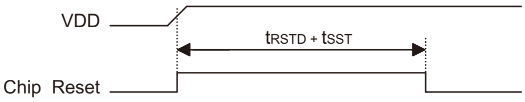
Note: ″u″ means unchanged

To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system awakes from the HALT state. Awaking from the HALT state, the SST delay is added.

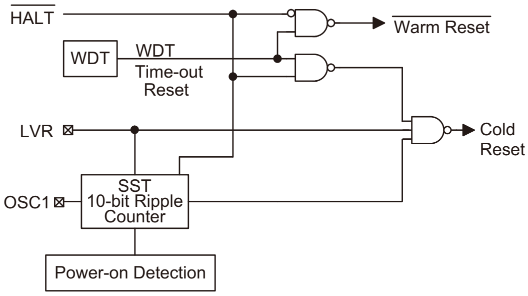
An extra SST delay is added during the power-up period and any wakeup from the HALT may enable only the SST delay.

The functional unit chip reset status is shown below.

Program Counter	000H
Interrupt	Disabled
Prescaler, Divider	Cleared
WDT, RTC, Time base	Cleared. After master reset, WDT starts counting
Timer/Event Counter	Off
Input/output ports	Input mode
Stack Pointer	Points to the top of the stack
Carrier Output	Floating state
SEG/COM outputs	High state



Reset Timing Chart



Reset Configuration

The chip reset register status is summarised in the following table:

Register	Reset (Power-on)	WDT Time-out Reset (Normal Operation)	LVR Reset (Normal Operation)	WDT Time-out (HALT)*
MP0	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu
MP1	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu
BP	---- ---0	---- ---0	---- ---0	---- ---u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RTCC	--00 0111	--00 0111	--00 0111	--uu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
CTRL	---- ---0	---- ---0	---- ---0	---- ---u
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	0000 1---	0000 1---	0000 1---	uuuu u---
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 111x	1111 111x	1111 111x	uuuu uuuu
SEGCR	0000 000x	0000 000x	0000 000x	uuuu uuuu
TSR0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TSR1	100- --00	100- --00	100- --00	uuu- --uu
CARL0	0000 0000	0000 0000	0000 0000	uuuu uuuu
CARL1	0000 0000	0000 0000	0000 0000	uuuu uuuu
CARH0	0000 0000	0000 0000	0000 0000	uuuu uuuu
CARH1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC1	--00 --00	--00 --00	--00 --00	--uu --uu
LCDC	---- --11	---- --11	---- --11	---- --uu

Note: ″*″ refers to warm reset

″u″ means unchanged

″x″ means unknown

″-″ means unimplemented

Timer/Event Counter

One timer/event counters are implemented in the devices. It contains an 8-bit programmable count-up counter.

The timer/event counter clock source may come from the system clock or system clock/4 or RTC time-out signal or external source. System clock source or system clock/4 is selected by option.

Using external clock input allows the user to count external events, measure time internals or pulse widths, or generate an accurate time base. While using the internal clock allows the user to generate an accurate time base.

There are two registers related to the timer/event counter, i.e., TMR (0DH) and TMRC (0EH). There are also two physical registers are mapped to TMR location; writing TMR places the starting value in the timer/event counter preload register, while reading it yields the contents of the timer/event counter. TMRC is timer/event counter control register used to define some options.

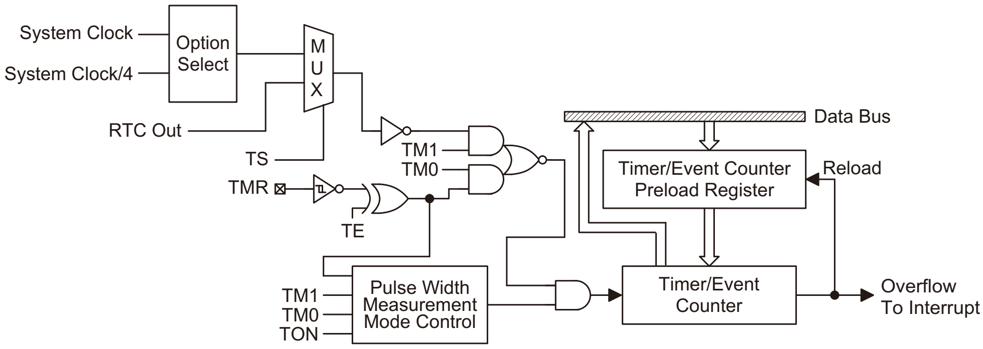
The TM0 and TM1 bits define the operation mode. The event count mode is used to count external events, which means that the clock source is from an external (TMR) pin. The timer mode functions as a normal timer with the clock source coming from the internal selected clock source. Finally, the pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR), and the counting is based on the internal 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. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag (TF;bit 6 of INTC0).

In the pulse width measurement mode with the values of the TON and TE bit equal to one, after the TMR has received a transient from low to high (or high to low if the TE bit is ″0″), it will start counting until the TMR returns to the original level and resets the TON. The measured result remains in the timer/event counter even if the activated transient occurs again. In other words, only one cycle measurement can be made until the TON is set. The cycle measurement will re-function as long as it receives further transient pulse. In this operation mode, the timer/event counter begins counting according not to the logic level but to the transient edges. In the case of counter overflows, the counter is reloaded from the timer/event counter preload register and issues an interrupt request, as in the other two modes, i.e., event and timer modes.

Bit No.	Label	Function
0~2	—	Unused bit, read as ″0″
3	TE	To define the TMR active edge of timer/event counter (0=active on low to high; 1=active on high to low)
4	TON	To enable/disable timer counting (0=disabled; 1=enabled)
5	TS	2 to 1 multiplexer control inputs to select the timer/event counter clock source (0=RTC outputs; 1= system clock or system clock/4)
6 7	TM0 TM1	To define the operating mode (TM1, TM0) 01=Event count mode (External clock) 10=Timer mode (Internal clock) 11=Pulse Width measurement mode (External clock) 00=Unused

TMRC (0EH) Register



Timer/Event Counter

To enable the counting operation, the Timer ON bit (TON: bit 4 of TMRC) should be set to 1. In the pulse width measurement mode, the TON is automatically cleared after the measurement cycle is completed. But in the other two modes, the TON can only be reset by instructions. The overflow of the timer/event counter is one of the wake-up sources. No matter what the operation mode is, writing a 0 to ETI disables the related interrupt service.

In the case of timer/event counter OFF condition, writing data to the timer/event counter preload register also reloads that data to the timer/event counter. But if the timer/event counter is turn on, data written to the timer/event counter is kept only in the timer/event counter preload register. The timer/event counter still continues its operation until an overflow occurs.

When the timer/event counter (reading TMR) is read, the clock is blocked to avoid errors. As this may results in a counting error, blocking of the clock should be taken into account by the programmer.

It is strongly recommended to load a desired value into the TMR register first, then turn on the related timer/event counter for proper operation. Because the initial value of TMR is unknown.

Due to the timer/event scheme, the programmer should pay special attention on the instruction to enable then disable the timer for the first time, whenever there is a need to use the timer/event function, to avoid unpredicatable result. After this procedure, the timer/event function can be operated normally.

Carrier Generator

The device provides a carrier generator of which the signal appears on the REM/REMDRV pin. The carrier output is generated by a 9-bit down-count counter enabled by a control bit together with a carrier signal generator.

- Timer Configuration

  This timer is an internal unit for creating a remote control transmission pattern. As shown, it consists of a 9-bit down-count counter (T8 to T0), a flag (T9) permitting the 1-bit timer output and a zero detector.

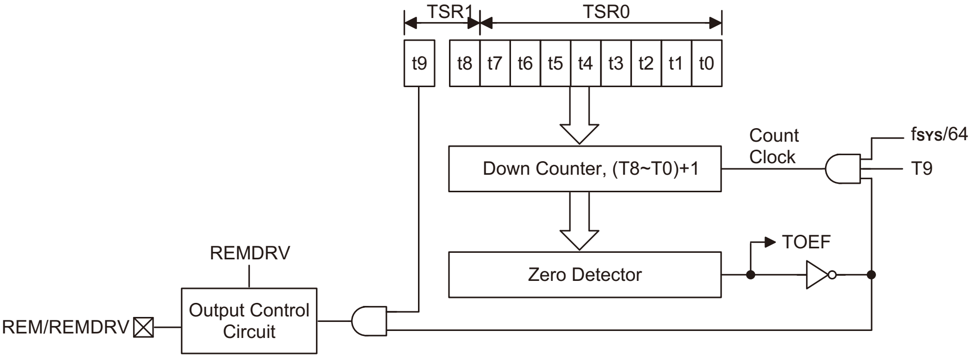
  There are two registers related to this timer known as TSR0 and TSR1 registers.

Bit No.	Label	Function
0~7	T7~T0	Timer bit 7~0

TSR0 Register

Bit No.	Label	Function
0	T8	Timer bit 8
1	T9	T9: Timer enable control 0: Disable 1: Enable
2~4	—	Unused bit, read as "0"
5	READYB	READYB: REMDRV output driver ready flag 0: REMDRV driver is ready for carrier output 1: REMDRV driver is not ready for carrier output This bit is used to indicate that whether the REMDRV output driver gets ready to deliver the carrier signal or not. When the REMDRV function is first enabled, including a wake-up from HALT instruction or an output function switch from REM to REMDRV mode, a certain period delay is necessary for the output driver to become stable before the REMDRV carrier signal is sent on the REMDRV pin. Users should make sure that the REMDRV output driver is ready by polling the READYB bit before the Timer is enabled. Note that in REM output function the READYB bit is not available and is always read as 0.
6	REMDRV	REM/REMDRV output function selection 0: REMDRV 1: REM
7	TOEF	Timer Operation End Flag 0: Timer operation is in progress 1: Timer operation is ended

TSR1 Register



Timer Configuration

- Timer Operation

The timer starts counting down when a value other than ″0″ is set for the down counter with a timer manipulation instruction. The timer manipulation instructions for making the timer start operation are shown below:

```
        MOV   A,XXH       ; Load T7~T0 into TSR0; XX = 00H ~ FFH
        MOV   TSR0,A
        MOV   A,01H       ; Load T8 into TSR1 and operate in REMDRV output mode
        MOV   TSR1,A

  LOOP: SZ    READYB      ; recommended to add the code to the program when a wake-up from HALT,
        JMP   LOOP        ; or an output function switch from REM to REMDRV mode

        SET   TSR1.1      ; after READYB=0, set T9 to 1 to enable the carrier timer
```

- Additional notes for the 9-bit timer
  - Writing to the TSR0 register will only put the written data to the TSR0 register (T7~T0) while writing to the TSR1 register (T8) will transfer the specified data together with the TSR0 register contents to the 9-bit Down Counter. The TOEF bit will be cleared after the data transferred from TSR1 and TSR0 registers to the Down Counter is completed and then wait until the TSR1 register bit 1 is set by user.
  - Setting the TSR1 register bit 1 to 1, the timer will start to count. The timer will stop when the counter content is equal to 0 and then the TOEF bit is set to 1.
  - If the TSR1 register bit 1 is cleared during the timer counting, the timer will be stopped. Once the TSR1 register bit 1 is set by a specific sequence, i.e. 1→0→1, the down counter will reload the data from T8~T0, and then the down counter begins counting down with the new load data.
  - If the TSR1 register bit 1 and the TOEF bit both are equal to 1, the timer can re-start, after a new data is written to the TSR0 and TSR1 register, i.e. T8~T0, in sequence.

Note: If the Down counter content is equal to 000H, set the T9 bit to start the timer counting, the timer will only count 1 step. The timer output time = 64/f_{SYS}. → [(0+1) × 64/f_{SYS}=64/f_{SYS}]

The down counter is decremented (−1) in the cycle of 64/f_{SYS}. If the value of the down counter becomes ″0″, the zero detector generates the timer operation end signal to stop the timer operation. At this time, the TOEF bit will be set to ″1″. The output of the timer operation end signal is continued while the down counter is ″0″ and the timer is stopped. The following relational expression applies between the timer′s output time and the down counter′s setting value.

Timer output time = (Set value+1) × 64/f_{SYS}

An example is shown below for f_{SYS}=4MHz.

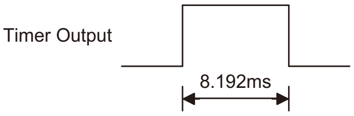
```
MOV   A,0FFH
MOV   TSR0, A
MOV   A,01H
MOV   TSR1, A
SET   TSR1.1
```

In the case above, the timer output time is as follows.

(Set value+1) × 64/f_{SYS}

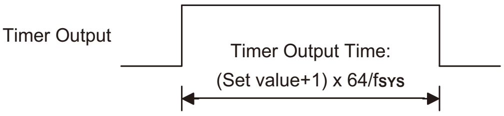
= (511+1) × 16µs

= 8.192ms



Setting the T9 bit channels the timer output to the REM/REMDRV pin. The REM/REMDRV pin will be a combination of the timer output and carrier signals.

Note: The timer output is shown as below if the high-level period setting modulo register named CARH bit 9 is set to 1.



Timer Output when Carrier is not Output

Carrier Output

- Carrier output generator

  The carrier generator consists of a 9-bit counter and two modulo registers known as the CARH and CARL registers respectively for setting the high-level and low-level periods.

  ◆ CARL0 Register

Bit	7	6	5	4	3	2	1	0
Name	CL7	CL6	CL5	CL4	CL3	CL2	CL1	CL0
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 7~0 **CL7~CL0**: Low level period modulo bit 7~0

◆ CARL1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CL9	CL8
R/W	—	—	—	—	—	—	R	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 unimplemented, read as ″0″

Bit 1 **CL9**: This bit is read only and fixed to ″0″

Bit 0 **CL8**: Low level period modulo bit 8

◆ CARH0 Register

Bit	7	6	5	4	3	2	1	0
Name	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
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 7~0 **CH7~CH0**: Low level period modulo bit 7~0

◆ CARH1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CH9	CH8
R/W	—	—	—	—	—	—	R	R/W
POR	—	—	—	—	—	—	1	0

Bit 7~2 unimplemented, read as ″0″

Bit 1 **CH9**: High level period modulo bit 9
If this bit is set to 1, the carrier signal will not be output.

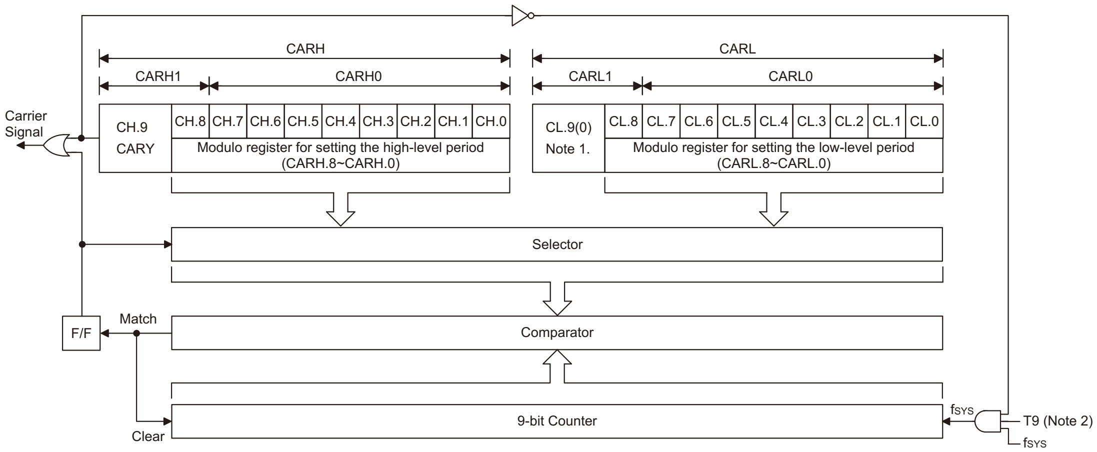
Bit 0 **CH8**: High level period modulo bit 8

The carrier duty ratio and carrier frequency can be determined by setting the high-level and low-level widths using the respective modulo registers.

CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0) are read and written using instructions.

Example:

```
    MOV    A, XXH         ; XXH = 00H~FFH
    MOV    CARL0, A
    MOV    A, XXH         ; XXH 01H, CL.8 (CARL1.0)
    MOV    CARL1, A
    MOV    A, XXH         ; XXH = 00H~FFH
    MOV    CARH0, A
    MOV    A, XXH         ; XXH 02H, CH.8 (CARH1.0)
    MOV    CARH1, A
    CLR    CARH1.1        ; The carrier is started by clearing CARH1.1 to 0
```



Configuration of Remote Controller Carrier Generator

Note: 1. The bit 9 of the modulo register for setting the low-level period (CARL) is fixed to 0.

2. T9: Flag that enables timer output (timer block, see Timer Configuration)

The values of CARH and CARL can be calculated from the following expressions.

CARL (CARL1.0, CARL0.7 ~ CARL0.0) = ($f_{SYS} \times (1-D) \times T) - 1$ ……… (1)

CARH (CARH1.0, CARH0.7 ~ CARH0.0) = ($f_{SYS} \times D \times T) - 1$ ………… (2)

(1) + (2) → CARL + CARH = ($f_{SYS} \times T) - 2$ → Actual Carrier Frequency = f_{SYS} / (CARL + CARH + 2)

D: Carrier duty ratio ($0 < D < 1$)

f_{SYS}: Input clock (MHz)

T: Carrier cycle (µs)

Ensure to input values in the range from 001H to 1FFH to the CARL and CARH registers.

If f_{SYS} = 4095kHz, Target f_C = 38kHz, T = 1 / f_C = 26.3157µs = $t_L + t_H$, duty = 1/3

CARL = (4.095M × (1 × 1/3) × 26.3157µs) × 1 = 70.842

select 71 = 47H, actual t_L = (71+1) / 4.095M = 17.58µs

CARH = (4.095M × 1/3 × 26.3157µs) × 1 = 34.921

select 35=23H, actual t_H = (35+1) / 4.095M = 8.79µs

For actual Carrier Frequency = f_{SYS} / (CARL + CARH +2)

So, actual f_C = f_{SYS} / (CARL + CARH +2) = 4095kHz / (71 + 35 + 2) = 37.917kHz

- Carrier output control

The remote controller carrier can be output from the REM/REMDRV pin by clearing to zero bit 9 (CARY) of the modulo register for setting the high-level period (CARH).

Be sure to set the timer operation after setting the CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0) values when performing a carrier output.

Note that a malfunction may occur if the values of the CARH and CARL registers are changed while the carrier is being output on the REM/REMDRV pin.

Executing the timer manipulation instruction starts the carrier output from the low level.

There is a dual function remote controller carrier output pin named REM/REMDRV. The selection of REM or REMDRV is determined by the TSR1 register bit 6. After a reset, the REM/REMDRV pin will be initiated to a REMDRV carrier output function and its output status will be in a floating condition. The generic structures of the REM or REMDRV function are illustrated in the accompanying diagram in the Input/Output Ports section. As the exact construction of the carrier output pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the remote carrier output pins.

The output from the REM/REMDRV pin is in accordance with the value of the CARH bit 9 (CARY) and the timer output enable flag T9 together with the value of the 9-bit down timer counter (T0 to T8).

CARH1.1	Timer Output Enable Flag (t9: TSR1.1)	9-bit Down Counter	REM Function (CMOS Output)	REMDRV Function (NMOS Output)
0	0	0	Low-level output	Floating output
0	0	Other than 0		
0	1	0	$64/f_{SYS}$ (with carrier output)	$64/f_{SYS}$ (with carrier output)
0	1	Other than 0	Carrier output (Note)	Carrier output
1	0	—	Low-level output	Floating output
1	1	—	High-level output	Low-level output

REM Pin Output Control

Note: The input value for CARH and CARL should be in the range from 001H to 1FFH.

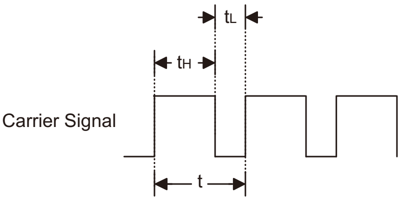
Caution: The CARH and CARL value must be set while the REM/REMDRV pin is at its inactive level.

Target		Setting		Actual			
f_C(kHz)	Duty	CARH (CARH 1, 0, CARH0.7~CARH0.0)	CARL (CARL 1,0, CARL0.7~CARL0.0)	t_H(μs)	t_L(μs)	T(μs)	f_C(kHzs)
36	1/3	25H	4BH	9.28	18.56	27.84	35.92
38	1/3	23H	47H	8.79	17.58	26.37	37.92
56	1/3	18H	2FH	6.11	11.72	17.83	56.10
56	1/2	23H	24H	8.79	9.04	17.83	56.10

Carrier Frequency Setting (f_{SYS}=4095kHz)

Target		Setting		Actual			
f_C(kHz)	Duty	CARH (CARH 1, 0, CARH0.7~CARH0.0)	CARL (CARL 1,0, CARL0.7~CARL0.0)	t_H(μs)	t_L(μs)	T(μs)	f_C(kHzs)
36	1/3	24H	49H	9.25	18.50	27.75	36.04
38	1/3	22H	45H	8.75	17.50	26.25	38.10
56	1/3	17H	2EH	6.00	11.75	17.75	56.34
56	1/2	23H	22H	9.00	8.75	17.75	56.34

Carrier Frequency Setting (f_{SYS}=4MHz)



Input/Output Ports

There are three 8-bit bidirectional input/output ports in the device, labeled as PA, PB and PC which are mapped to [12H], [14H], [16H] of the RAM respectively. Each bit of PA, PB and PC can be selected as NMOS output or Schmitt trigger with pull-high resistor by software instruction. Note that PB7 is without pull-high resistor.

When PA, PB and PC for the input operation, these ports are non-latched, that is, the inputs should be ready at the T2 rising edge of the instruction ″MOV A, [m]″ (m=12H, 14H or 16H). For PA, PB and PC output operation all data are latched and remain unchanged until the output latch is rewritten.

When the PA, PB and PC are used for input operation, it should be noted that before reading data from pads, a ″1″ should be written to the related bits to disable the NMOS device. That is, the instruction ″SET [m].i″ (i=0~7, m=12H, 14H or 16H) is executed first to disable the related NMOS device, and then use the instruction ″MOV A, [m]″ to get stable data.

After chip reset, PA, PB0~6 and PC remain at a high level input line and PB7 remains at a high impedance input line. Each bit of PA, PB and PC output latches can be set or cleared by the ″SET [m].i″ and ″CLR [m].i″ (m=12H , 14H or 16H) instructions respectively.

Some instructions first input data and then follow the output operations. For example, ″SET [m].i″, ″CLR [m]″, ″CPL [m]″, ″CPLA [m]″ read the entire port states into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or to the accumulator.

Each line of PA and PB has a wake-up capability to the device by configuration option.

Pins PC1~PC7 are pin-shared with SEG1~SEG7 and are selected by the LCD Segment Control Register named SEGCR.

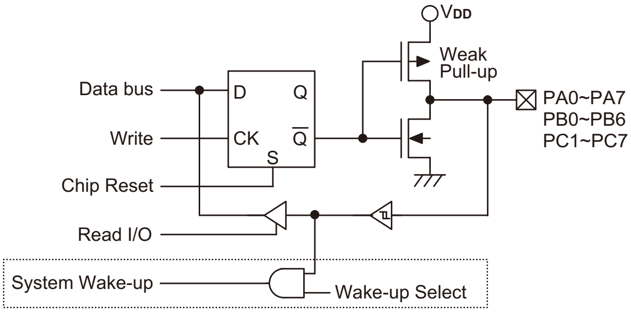
- **I/O Resistor Lists**

Register Name	Bit							
	7	6	5	4	3	2	1	0
SEGCR	SEG7C	SEG6C	SEG5C	SEG4C	SEG3C	SEG2C	SEG1C	Reserved
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	Reserved

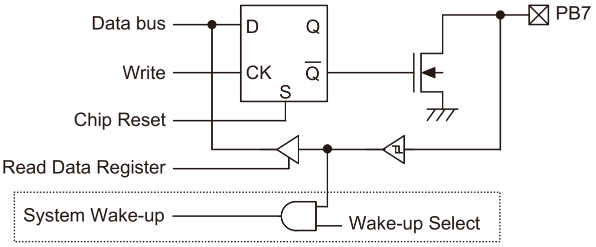
- SEGCR Register

Bit	7	6	5	4	3	2	1	0
Name	SEG7C	SEG6C	SEG5C	SEG4C	SEG3C	SEG2C	SEG1C	Reserved
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	x

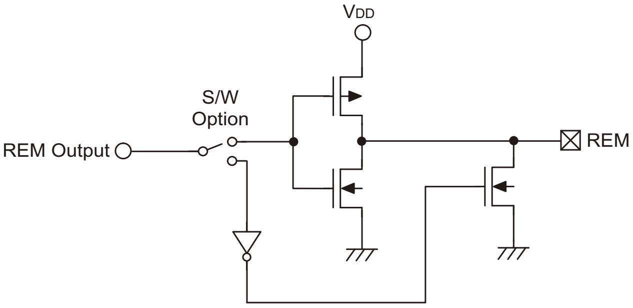
Bit 7~0 **SEG7C~SEG1C**: Segment 7~ Segment 1 selection bits
0: I/O
1: Segment output



Note that the pin-shared functions are not shown here.
PA, PB0~PB6 and PC Input/Output Ports



PB7 Input/Output Pin



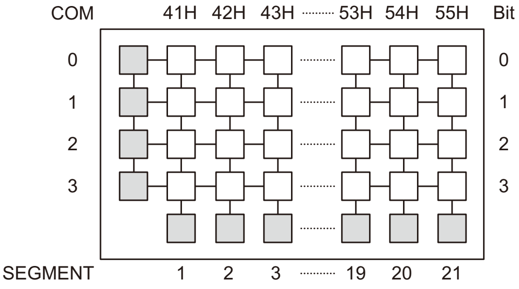
Note that the S/W Option is the TSR1 register bit 6
REM/REMDRV Pin Structure

LCD Driver

LCD Display Memory

The device provides an area of embedded data memory for LCD display. This area is located from address 41H to 55H of the RAM at Bank 1. The Bank pointer (BP; located at 04H of the RAM) is the switch between the RAM and the LCD display memory. When the BP is set as ″1″, any data written into 41H~55H will affect the LCD display. When the BP is cleared to ″0″, any data written into 41H~55H means to access the general purpose data memory.

The LCD display memory can be read and written to only by indirect addressing mode using MP1. When a data is written into the display data area, it is automatically read by the LCD driver which then generates the corresponding LCD driving signals. To turn the display on or off, a ″1″ or a ″0″ is written to the corresponding bit of the display memory, respectively. The figure illustrates the mapping between the display memory and LCD pattern for the device.



Display Memory

LCD Driver Output

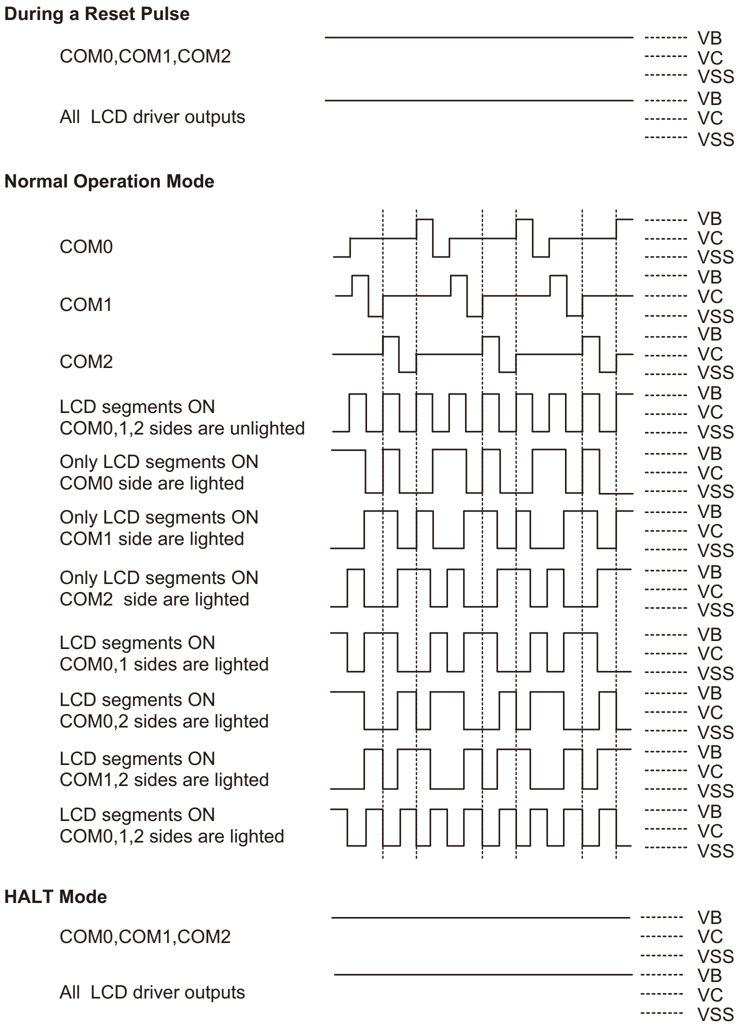
The output number of the LCD driver device can be 21×2, 21×3 or 20×4 by configuration option. The LCD driver bias type is ″C″ type only. A capacitor mounted between C1 and C2 pins is needed. If 1/2 bias is selected, a capacitor mounted between VC pin and ground is required. If 1/3 bias is selected, two capacitors are needed for VA and VC pins. All the capacitance of capacitors used for LCD bias generator is suggested to use the 0.1μF. The relationships between LCD bias types, bias levels and VA and VC connection are listed in the table.

Bias Level	Bias Type	C1/C2	VA	VC
1/2	C	0.1μF	x	0.1μF
1/3	C	0.1μF	0.1μF	0.1μF

There is a clock source needed for the LCD driver. The LCD clock source comes from the general purpose prescaler and is decided by configuration options. The LCD clock frequency should be selected as near to 4kHz either from 32768Hz LXT or LIRC or $f_{SYS}/4$ clock source.

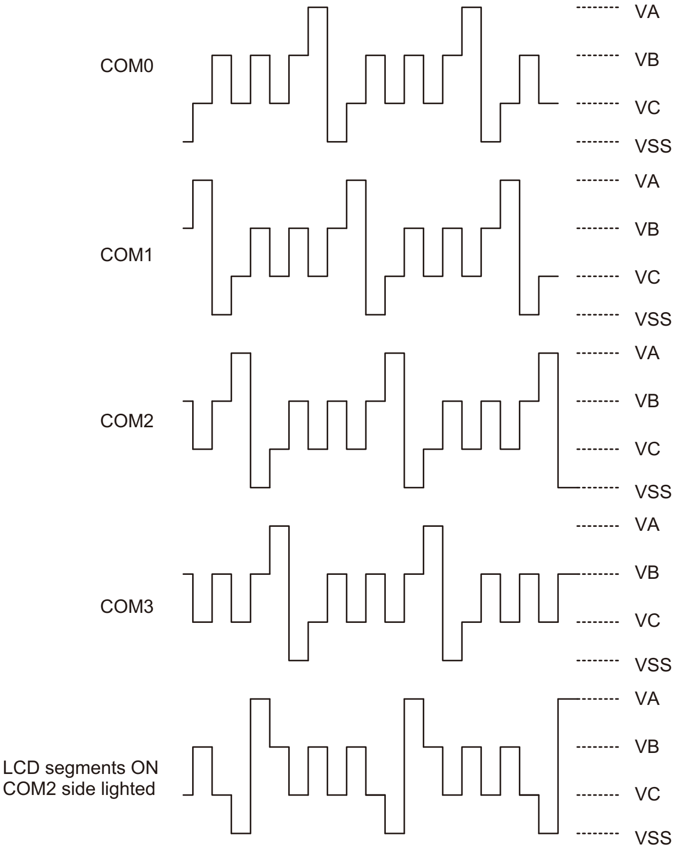
The options of LCD clock frequency are listed in the following table.

f_S Clock Source	LCD Clock Selection
LIRC oscillator	$f_{LIRC}/2^3$
LXT oscillator	$f_{LXT}/2^3$
$f_{SYS}/4$	$f_{SYS}/2^4$~$f_{SYS}/2^{10}$



Note: "*" Omit the COM2 signal, if the 1/2 duty LCD is used.
VB=VA=VDD, VC=VDDx1/2

LCD Driver Output (1/3 Duty, 1/2 Bias, C Type)



Note: VA=VDDx1.5, VB=VDD, VC=VDDx1/2

LCD Driver Output (1/4 Duty, 1/3 Bias, C Type)

LCD/RTC OSC Control Register

Two bit in (1FH) are for controlling LCD and RTC OSC.

Bit No.	Label	Function
0	RTCEN	Controls RTC OSC enable (1=enabled; 0=;disabled)
1	LCDEN	Controls LCD enable (1=enabled; 0=disabled)
2~7	—	Unused bit, read as ″0″

LCDC (1FH) Register

LCDEN and RTCEN may decide LCD and RTC On/Off condition on normal operation.

f_S Clock Source	LCD/RTC Control Bits			
	LCDEN, RTCEN=0, 0	LCDEN, RTCEN=0, 1	LCDEN, RTCEN=1, 0	LCDEN, RTCEN=1, 1
$f_{SYS}/4$	LCD off, RTC off	LCD off, RTC off	LCD off, RTC off	LCD off, RTC off
WDT OSC	LCD off, RTC off	LCD off, RTC off	LCD on, RTC off	LCD on, RTC off
RTC OSC	LCD off, RTC on	LCD off, RTC on	LCD on, RTC on	LCD on, RTC on

Low Voltage Reset/Detector Functions

There is a low voltage detector (LVD) and a low voltage reset circuit (LVR) implemented in the microcontroller. LVR is always enabled except in HALT mode. LVD can be enabled/disabled by options. Once the LVD options is enabled, the user can use the RTCC.3 to enable/disable (1/0) the LVD circuit and read the LVD detector status (0/1) from RTCC.5; otherwise, the LVD function is disabled.

The RTCC register definitions are listed below.

Bit No.	Label	Function
0~2	RT0~RT2	8 to 1 multiplexer control inputs to select the real clock prescaler output
3	LVDC*	LVD enable/disable (1/0)
4	QOSC	32768Hz OSC quick start-up oscillating 0/1: quickly/slowly start
5	LVDO	LVD detection output (1/0) 1: low voltage detected, read only
6, 7	—	Unused bit, read as ″0″

RTCC (09H) Register

Once the LVD function is enabled the reference generator should be enabled; otherwise the reference generator is controlled by LVR Enable/Disable. The relationship among LVR and LVD options and LVDC are as shown.

LVDC can read/write, LVDO is read only.

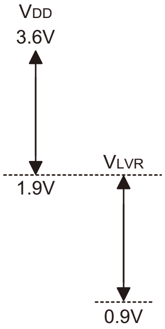
LVD	LVR	LVDC	VREF Generator	LVR Comparator	LVD Comparator
Enable	Enable	On	Enable	Enable	Enable
Enable	Enable	Off	Enable	Enable	Disable
Enable	Disable	On	Enable	Disable	Enable
Enable	Disable	Off	Disable	Disable	Disable
Disable	Enable	X	Enable	Enable	Disable

The microcontroller provides a low voltage reset circuit in order to monitor the supply voltage of the device. If the supply voltage of the device is within the range 0.9V~V_{LVR}, such as might happen when changing a battery, the LVR will automatically reset the device internally. During a HALT state, LVR is disabled.

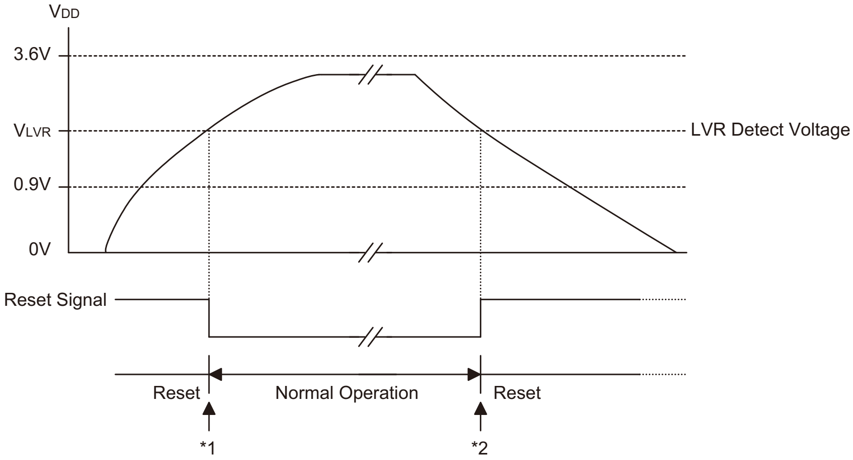
The LVR includes the following specifications:

- The low voltage (0.9V~V_{LVR}) state must exists for more than 1ms, while the other circuits remain in their original state. If the low voltage state does not exceed 1ms, the LVR will ignore it and do not perform a reset function.
- The LVR uses the ″OR″ function with the power-on reset signal to perform a chip reset.

The relationship between V_{DD} and V_{LVR} is shown below.



Note: V_{OPR} is the voltage range for proper chip operation at 4MHz system clock.



Low Voltage Reset

Note: ″*1″ To make sure that the system oscillator has stabilized, the SST provides an extra delay of 1024 system clock pulses before entering the normal operation.

″*2″ Low voltage state has to be maintained for over 1ms, then after a 1ms delay the device enters the reset mode.

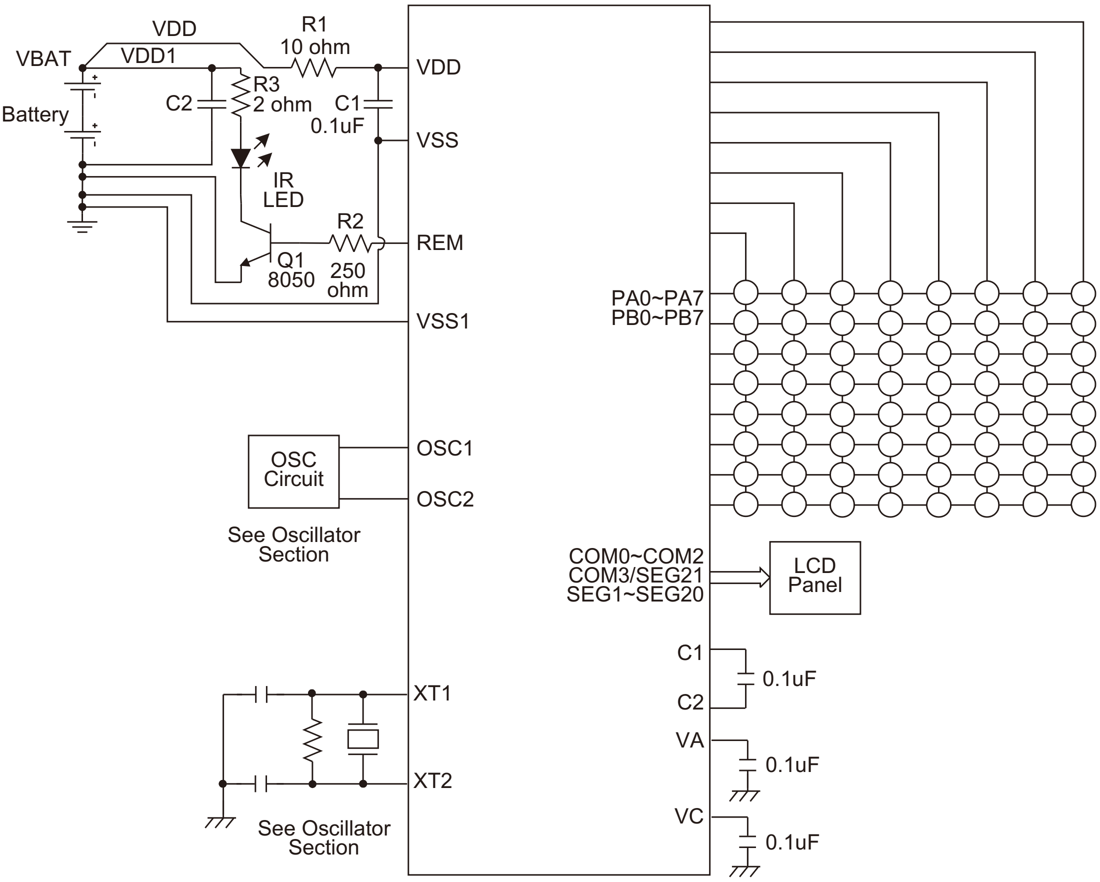
Options

The following table shows all kinds of options in the micro-controller. All of the options must be defined to ensure proper system functioning.

Item	Options
I/O Options	
1	PA0~PA7 wake-up: Enabled or Disabled
2	PB0~PB7 wake-up: Enabled or Disabled
Oscillator Options	
3	System oscillator selection - f_{SYS}: XTAL oscillator without internal feedback resistor XTAL oscillator with internal feedback resistor Internal 4095kHz RC oscillator
4	f_S internal clock source: f_{SYS} /4 or LXT OSC or LIRC OSC
Interrupt Options	
5	INT0 function : Disable, rising edge, falling edge or both edges
6	INT1 function : Disable, rising edge, falling edge or both edges
Timer/Event Counter Options	
7	Timer/Event Counter (f_{TMR}) clock source: $f_{SYS}/4$ or f_{SYS}
Time Base Options	
8	Time Base frequency (f_{TB}): $f_S/2^{12}$ ~ $f_S/2^{15}$
LVD Option	
9	LVD Low Voltage Detect : enable or disabled
LCD Options	
10	f_{LCD}: LCD frequency (typical 4kHz): $f_S/2^2$ ~ $f_S/2^8$
11	LCD duty: 1/2, 1/3, 1/4 duty
12	LCD bias: 1/2, 1/3 bias
I/O Pin Option	
13	I/O pin or XT1/XT2 pin

Application Circuits

The following application circuit shows the situation when a transistor is added to IR driver circuit. Here the MCU REM function must be enabled.

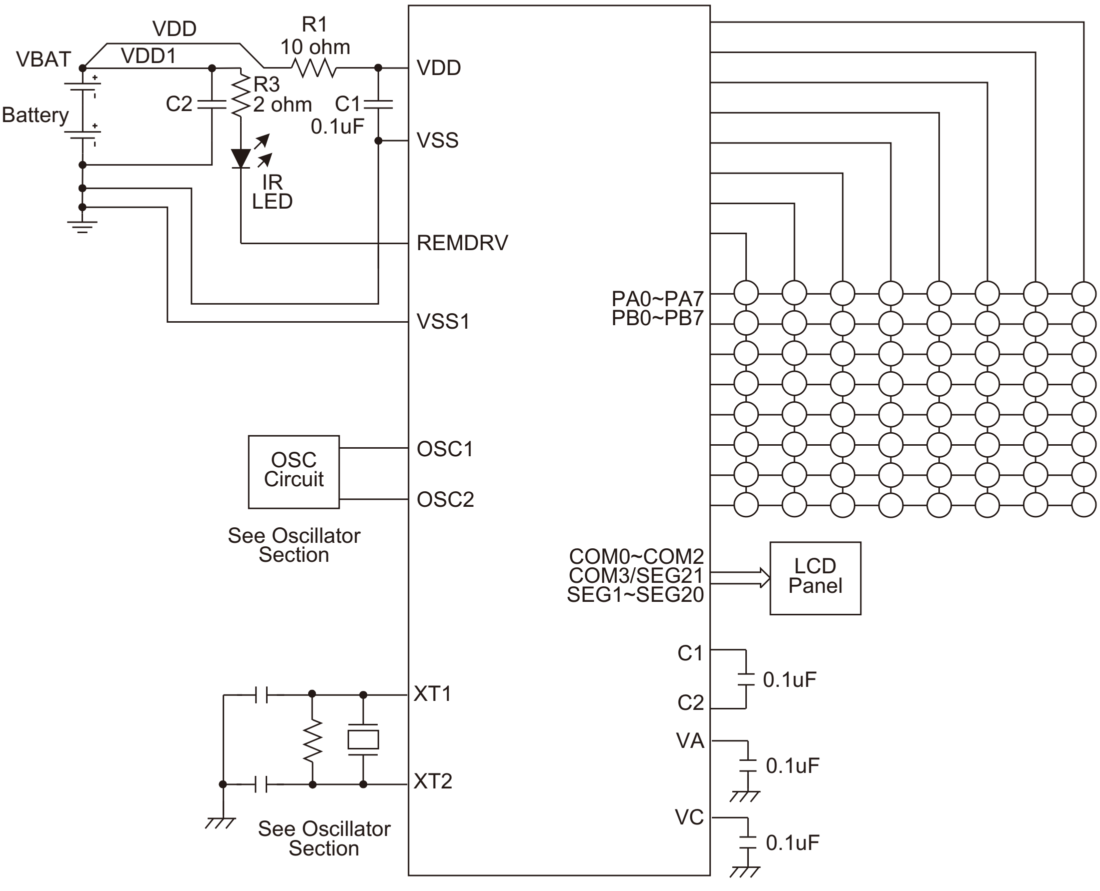


HT49RA0-6

Note:

1. The values of R1 and C2 should be selected in consultation with the actual application, R1=10Ω, C1=0.1μF, C2=200~330μF are suggested values.
2. To obtain a better frequency stability and longer transmission distances, C2=330μF is a recommended value. The frequency stability may be different and the transmission distance may be shorter if a value other than 330μF is used.
3. VSS, VSS1, C2 and Q1 must be connected to the power GND terminal.
4. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS on the PCB .
5. The C1 (0.1μF) decoupling capacitor should be located as close to the VDD and VSS pins as possible.
6. R1 and C1 should be located as close to the VDD pin as possible
7. VDD and VDD1 must be connected to the power VBAT terminal.
8. The values of R1 and C2 should be selected in consultation with the actual application
9. It should to be noted that when programming the device, the HT49RA0-6 writer type is the e-Writer PRO, which when used together with the e-Socket, can ensure that the HIRC oscillator frequency will have a tolerance within 1% in the actual application circuit.

The following application circuit shows the situation when the internal IR driver circuit is used. Here the MCU REMDRV function must be enabled.



HT49RA0-6

Note:

1. The values of R1 and C2 should be selected in consultation with the actual application, R1=10Ω, C1=0.1μF, C2=47~100μF are suggested values.
2. To obtain a better frequency stability and longer transmission distances, C2=100μF is a recommended value. The frequency stability may be different and the transmission distance may be shorter if a value other than 100uF is used.
3. VSS, VSS1, C2 and Q1 must be connected to the power GND terminal.
4. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS on the PCB.
5. The C1 (0.1μF) decoupling capacitor should be located as close to the VDD and VSS pins as possible.
6. R1 and C1 should be located as close to the VDD pin as possible
7. VDD and VDD1 must be connected to the power VBAT terminal.
8. The values of R1 and C2 should be selected in consultation with the actual application
9. It should to be noted that when programming the device, the HT49RA0-6 writer type is the e-Writer PRO, which when used together with the e-Socket, can ensure that the HIRC oscillator frequency will have a tolerance within 1% in the actual application circuit.

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

Mnemonic	Description	Cycles	Flag Affected
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1^{Note}	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1^{Note}	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1^{Note}	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1^{Note}	C
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1^{Note}	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1^{Note}	None
SET [m].i	Set bit of Data Memory	1^{Note}	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1^{Note}	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1^{note}	None
SZ [m].i	Skip if bit i of Data Memory is zero	1^{Note}	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1^{Note}	None
SIZ [m]	Skip if increment Data Memory is zero	1^{Note}	None
SDZ [m]	Skip if decrement Data Memory is zero	1^{Note}	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1^{Note}	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1^{Note}	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	2^{Note}	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2^{Note}	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1^{Note}	None
SET [m]	Set Data Memory	1^{Note}	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	1^{Note}	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note:
1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the ″CLR WDT1″ and ″CLR WDT2″ instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both ″CLR WDT1″ and ″CLR WDT2″ instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a 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 WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

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

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF

CPL [m] Complement Data Memory

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

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation $[m] \leftarrow [m] - 1$

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

HALT Enter power down mode

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

Operation $TO \leftarrow 0$
$PDF \leftarrow 1$

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s)
C

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

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

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

Affected flag(s)
C

RR [m] Rotate Data Memory right

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

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

Affected flag(s)
None

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

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

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

Affected flag(s)
None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s)
C

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

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

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

Affected flag(s)
C

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

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

Operation $\mathrm{ACC} \leftarrow \mathrm{ACC} - [m] - \overline{C}$

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 $[m] \leftarrow \mathrm{ACC} - [m] - \overline{C}$

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 instruction.

Operation $\mathrm{ACC} \leftarrow [m] - 1$
Skip if ACC = 0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation $[m] \leftarrow \mathrm{FFH}$

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation $[m].i \leftarrow 1$

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.

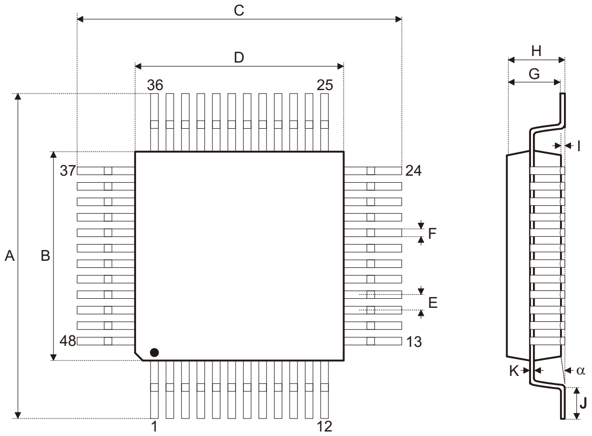
Operation ACC ← ACC ″XOR″ x

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.

48-pin LQFP (7mm×7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.350	—	0.358
B	0.272	—	0.280
C	0.350	—	0.358
D	0.272	—	0.280
E	—	0.020	—
F	—	0.008	—
G	0.053	—	0.057
H	—	—	0.063
I	—	0.004	—
J	0.018	—	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	8.90	—	9.10
B	6.90	—	7.10
C	8.90	—	9.10
D	6.90	—	7.10
E	—	0.50	—
F	—	0.20	—
G	1.35	—	1.45
H	—	—	1.60
I	—	0.10	—
J	0.45	—	0.75
K	0.10	—	0.20
α	0°	—	7°

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

Holtek Semiconductor Inc. (Taipei Sales Office)
4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan
Tel: 886-2-2655-7070
Fax: 886-2-2655-7373
Fax: 886-2-2655-7383 (International sales hotline)

Holtek Semiconductor (China) Inc.
Building No. 10, Xinzhu Court, (No. 1 Headquarters), 4 Cuizhu Road, Songshan Lake, Dongguan, China 523808
Tel: 86-769-2626-1300
Fax: 86-769-2626-1311

Holtek Semiconductor (USA), Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holtek.com

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek′s products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at http://www.holtek.com.tw.