R-F Type Low Voltage 8-bit Mask MCU

HT47C07L/HT47C08L

Revision: V1.10 Date: December 23, 2014

www.holtek.com

Table of Contents

Features

- Operating voltage: 1.2V~2.2V
- Up to 21 bidirectional I/O lines
- On-chip 32kHz~128kHz Internal RC oscillator
- Up to two RC type A/D converters
- Watchdog Timer
- Up to 2K×16 program memory ROM
- Up to 96×8 data memory RAM
- Time Base Function
- One buzzer output (BZ, $\overline{BZ}$)
- One EL carrier output
- One LCD driver with up to 21×3 segments
- HALT function and wake-up feature reduce power consumption
- Four-level subroutine nesting
- Low Voltage Detect Function
- Bit manipulation instruction
- 16-bit table read instruction
- Up to 31μs instruction cycle with 128kHz system clock
- 63 powerful instructions
- All instructions in one or two machine cycles
- 48-pin LQFP package types

General Description

The HT47C07L and HT47C08L are 8-bit, high performance RISC architecture microcontroller devices specifically designed for applications that interface directly to analog signals, such as those from sensors. Its single cycle instruction and two-stage pipeline architecture make it suitable for high speed applications.

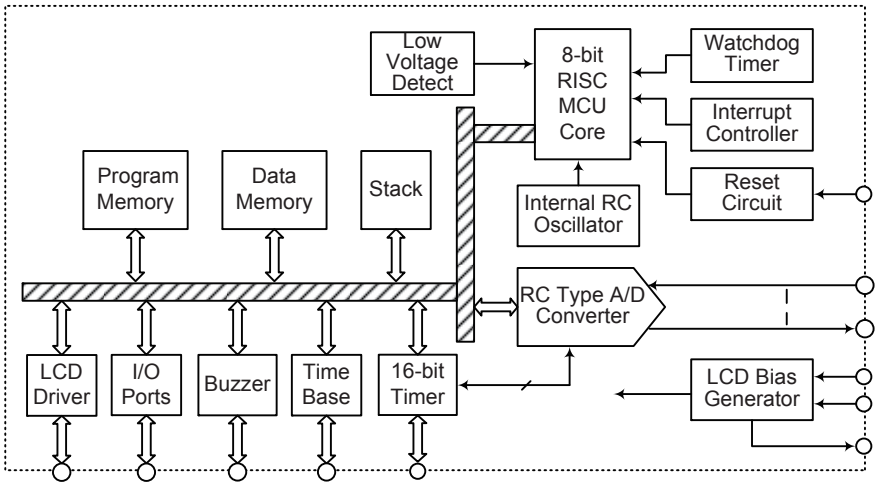
The advantages of low power consumption, I/O flexibility, timer functions, oscillator options, RC type A/D converter, LCD driver, HALT and wake-up functions, enhance the versatility of these device to suit a wide range of Resistor to Frequency application possibilities such as sensor signal processing, remote metering, and particularly suitable for use as clinical thermometer MCU device.

Selection Table

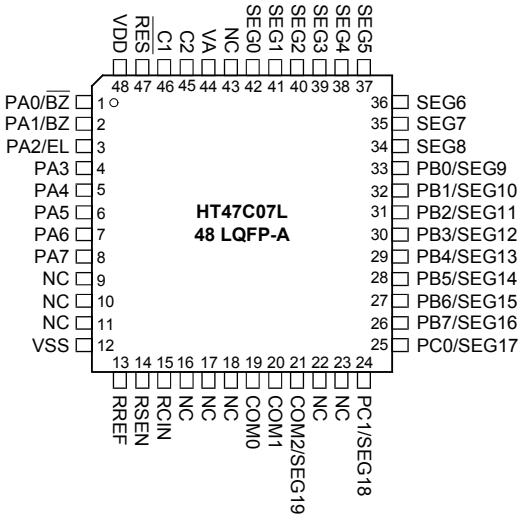
Most features are common to all devices. The main features distinguishing them are Memory capacity, I/O count, RC Type A/D Converter channels, LCD Driver Outputs and package types. The following table summarises the main features of each device.

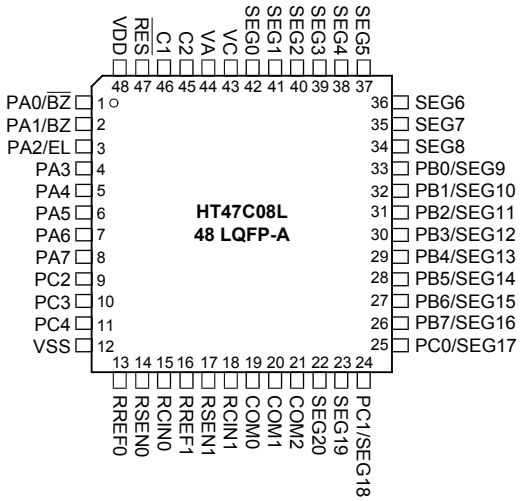
Part No.	Program Memory	Data Memory	I/O	EL Carrier Output	RC Type A/D Converter	16-bit Timer	LCD Driver	Time Base	Stacks	Package
HT47C07L	1K×16	48×8	18	1	1	1	19×3 or 20×2	1	4	48LQFP
HT47C08L	2K×16	96×8	21	1	2	1	21×3	1	4	48LQFP

Block Diagram



Pin Assignment





Pin Descriptions

HT47C07L

Pin Name	I/O	Option	Pin-Shared Mapping
$\overline{RES}$	I	—	Schmitt trigger reset input. Active low.
PA0/BZ PA1/$\overline{BZ}$	I/O I/O	Wake-up, Pull-high or None BZ or $\overline{BZ}$	Bidirectional 2-bit input/output port. Each bit can be a wake-up input. The PA0 and PA1 are pin-shared with the BZ and $\overline{BZ}$, respectively. Once the PA0 and PA1 are selected as buzzer driving outputs, the output signals come from an internal buzzer clock generator. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (mask option).
PA2/EL	I/O	Wake-up, Pull-high or None EL	Bidirectional 1-bit input/output port. This bit can be a wake-up input. The PA2 is pin-shared with the EL carrier output. Once the PA2 is selected as EL carrier output, the output signal comes from an internal EL carrier clock generator. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (mask option).
PA3~PA7	I/O	Wake-up, Pull-high or None	Bidirectional 5-bit input/output port. Each bit can be a wake-up input. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (mask option).
PB0/SEG9~ PB7/SEG16	I/O	I/O or SEG	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (configuration option). These pins are used as the LCD driver outputs for LCD panel segments. Configuration options can select each pin to be used as either an I/O pin or a segment driver output.
PC0/SEG17~ PC1/SEG18	I/O	I/O or SEG	Bidirectional 2-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (configuration option). These pins are used as the LCD driver outputs for LCD panel segments. Configuration options can select each pin to be used as either an I/O pin or a segment driver output.
SEG0~SEG8	O	—	LCD driver output for LCD panel segments.
COM0~COM1, COM2/SEG19	O	1/2 or 1/3 Duty	COM0~COM1 are the LCD common outputs. The 1/3 LCD duty cycle mask option will determine whether pin COM2/SEG19 is configured as a SEG19 segment output driver or as a COM2 common output driver for the LCD panel.
VA, C1, C2	—	—	VA: LCD power supply voltage, connect a capacitor between VA and VSS. C1, C2: Switching pins for VA. Connect a capacitor between C1 and C2.
RCIN	I	—	RC type A/D converter input pin for RC oscillation.
RREF	O	—	RC type A/D converter output pin for reference resistor oscillation.
RSEN	O	—	RC type A/D converter output pin for sensor resistor oscillation.
VDD	P	—	Positive power supply.
VSS	P	—	Negative power supply, ground.

HT47C08L

Pin Name	I/O	Option	Pin-Shared Mapping
$\overline{RES}$	I	—	Schmitt trigger reset input. Active low.
PA0/BZ PA1/$\overline{BZ}$	I/O I/O	Wake-up, Pull-high or None BZ or $\overline{BZ}$	Bidirectional 2-bit input/output port. Each bit can be a wake-up input. The PA0 and PA1 are pin-shared with the BZ and $\overline{BZ}$, respectively. Once the PA0 and PA1 are selected as buzzer driving outputs, the output signals come from an internal buzzer clock generator. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (mask option).
PA2/EL	I/O	Wake-up, Pull-high or None EL	Bidirectional 1-bit input/output port. This bit can be a wake-up input. The PA2 is pin-shared with the EL carrier output. Once the PA2 is selected as EL carrier output, the output signal comes from an internal EL carrier clock generator. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (mask option).
PA3~PA7	I/O	Wake-up, Pull-high or None	Bidirectional 5-bit input/output port. Each bit can be a wake-up input. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (mask option).
PC2~PC4	I/O	—	Bidirectional 3-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (mask option).
PB0/SEG9~ PB7/SEG16	I/O	I/O or SEG	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (configuration option). These pins are used as the LCD driver outputs for LCD panel segments. Configuration options can select each pin to be used as either an I/O pin or a segment driver output.
PC0/SEG17~ PC1/SEG18	I/O	I/O or SEG	Bidirectional 2-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (configuration option). These pins are used as the LCD driver outputs for LCD panel segments. Configuration options can select each pin to be used as either an I/O pin or a segment driver output.
SEG0~SEG8, SEG19~20	O	—	LCD driver output for LCD panel segments.
COM0~COM2	O	—	COM0~COM2 are the LCD common outputs.
VA, VC, C1, C2	—	—	VA, VC: LCD power supply voltage, connect a capacitor between VA and VSS and between VC and VSS, respectively. C1, C2: Switching pins for VA and VC. Connect a capacitor between C1 and C2.
RCIN0	I	—	RC type A/D converter input pin0 for RC oscillation.
RREF0	O	—	RC type A/D converter output pin0 for reference resistor oscillation.
RSEN0	O	—	RC type A/D converter output pin0 for sensor resistor oscillation.
RCIN1	I	—	RC type A/D converter input pin1 for RC oscillation.
RREF1	O	—	RC type A/D converter output pin1 for reference resistor oscillation.
RSEN1	O	—	RC type A/D converter output pin1 for sensor resistor oscillation.
VDD	P	—	Positive power supply.
VSS	P	—	Negative power supply, ground.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}−0.3V to 2.5V
Input Voltage ..... V_{SS}−0.3V to V_{DD}+0.3V
Storage Temperature..... -50˚C to 125˚C
Operating Temperature..... -10˚C to 50˚C

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	1.2	—	2.2	V
V_A	LCD Voltage	—	1/2 bias	—	V_{DD}×2	—	V
V_B	LCD Voltage	—	1/2 bias	—	V_{DD}	—	V
V_{LVD1}	Low Voltage Detector Voltage	—	Mask Option	1.25	1.3	1.35	V
V_{LVD2}	Low Voltage Detector Voltage	—	Mask Option	1.22	1.27	1.32	V
V_{LVD3}	Low Voltage Detector Voltage	—	Mask Option	1.20	1.25	1.30	V
I_{DD1}	Operating Current	1.5V	No load, f_{SYS}=32kHz, A/D off, LVD disabled	—	—	5	μA
I_{DD2}	Operating Current	1.5V	No load, f_{SYS}=32kHz, A/D on, LVD disabled *R=30kΩ, *C=2200pF	—	—	45	μA
I_{DD3}	Operating Current	1.5V	No load, f_{SYS}=128kHz, A/D off, LVD disabled	—	—	20	μA
I_{DD4}	Operating Current	1.5V	No load, f_{SYS}=128kHz, A/D on, LVD disabled *R=30kΩ, *C=2200pF	—	—	60	μA
I_{LVD}	LVD Current	1.5V	LVD enabled	—	50	100	μA
I_{STB1}	Standby Current (LVD Disabled, LCD Off, System OSC off)	1.5V	No load, system HALT A/D Off, LVD disabled	—	0.1	1	μA
I_{STB2}	Standby Current (LCD On)	1.5V	No load, system HALT f_{SYS}=32kHz, A/D Off, LVD disabled	—	—	3	μA
I_{STB3}	Standby Current (LCD On)	1.5V	No load, system HALT f_{SYS}=128kHz, A/D Off, LVD disabled	—	—	13	μA
V_{IL1}	Input Low Voltage for I/O Ports	—	—	0	—	0.3V_{DD}	V
V_{IH1}	Input High Voltage for I/O Ports	—	—	0.8V_{DD}	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	0.4V_{DD}	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	0.9V_{DD}	—	V_{DD}	V
I_{OL1}	I/O Port Sink Current (PA, PB, PC)	1.5V	V_{OL}=0.1V_{DD}	0.5	0.8	—	mA
I_{OH1}	I/O Port Source Current (PA, PB, PC)	1.5V	V_{OH}=0.9V_{DD}	-0.3	-0.6	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OL2}	I/O Port Sink Current (RREFn, RSENn)	1.5V	V_{OL}=0.1V_{DD}	4	7	—	mA
I_{OH2}	I/O Port Source Current (RREFn, RSENn)	1.5V	V_{OH}=0.9V_{DD}	-3	-5	—	mA
I_{OL3}	LCD common and segment Sink Current	1.5V	V_{OL}=0.1V_{DD}	50	100	—	μA
I_{OH3}	LCD common and segment Source Current	1.5V	V_{OH}=0.9V_{DD}	-50	-100	—	μA
R_{PH}	Pull-high Resistance of I/O Ports	1.5V	—	75	150	300	kΩ

Note: *R stands for the RC type A/D converter resistance
*C stands for the RC type A/D converter capacitance

A.C. Characteristics

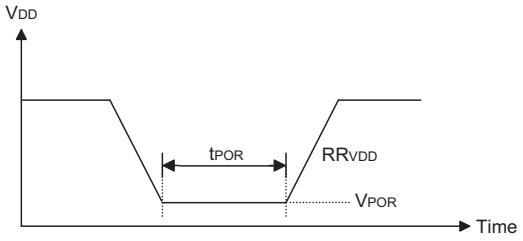
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{LIRC2}	LIRC2 Oscillator Clock	1.5V	—	102.4	128.0	153.6	kHz
t_{ONLIRC}	Stable Time for LIRC2 Oscillator	1.5V	—	—	—	1	ms
f_{SYS}	System Clock	1.5V	—	25	—	154	kHz
t_{RES}	External Reset Low Pulse Width	1.5V	—	100	—	—	μs
f_{AD}	A/D type RC oscillation frequency	1.5V	—	—	—	50	kHz
f_{LCD}	LCD Driver Clock Frequency	—	User select by configuration option	—	4	—	kHz

Power-on Reset Electrical Characteristics

Ta=25°C

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

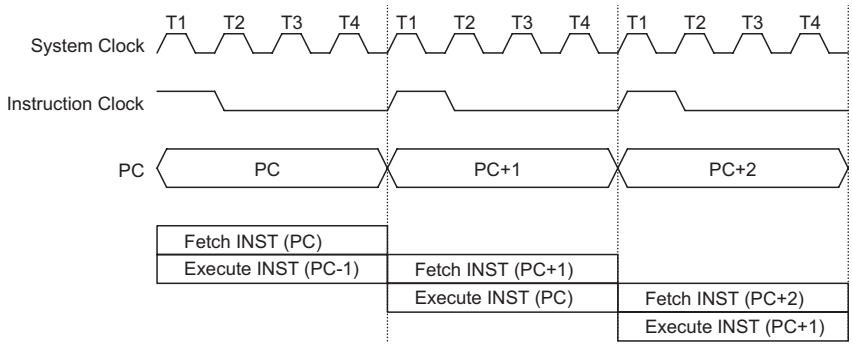


System Architecture

Clocking and Pipelining

The system clock is derived from an internal RC oscillator and is internally divided into four non-overlapping clocks (T1, T2, T3 and T4). 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. This pipelining scheme ensures that instructions are effectively executed in one cycle. Exceptions to this are instructions that change the contents of the program counter, such as subroutine calls or jumps, in which case, two cycles are required to complete the instruction.



Execution Flow

Program Counter – PC

The 10-bit and 11-bit program counter (PC) controls the sequence in which the instructions stored in the program ROM are executed and its contents specify a maximum of 1024 addresses for HT47C07L and 2048 addresses for HT47C08L respectively.

After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

When executing a jump instruction, conditional skip execution, loading PCL register, subroutine call, initial reset, internal interrupt, external interrupt or return from subroutine, etc., the microcontroller manages program control by loading the address corresponding to each instruction.

For conditional skip instructions, once the condition has been met, the next instruction, which has already been fetched during the current instruction execution, is discarded and a dummy cycle replaces it while the proper instruction is obtained. Otherwise proceed with the next instruction.

The lower byte of the program counter (PCL) is available for program control and is a readable and writeable register (06H). Moving data into the PCL performs a short jump. The destination will be within 256 locations.

When a control transfer takes place, an additional dummy cycle is required.

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
Timer/event Counter Interrupt	0	0	0	0	0	0	0	0	1	0	0
Time Base Interrupt	0	0	0	0	0	0	0	1	0	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

For HT47C07L, the program counter is 10 bits, i.e. bit9~bit0.

For HT47C08L, the program counter is 11 bits, i.e. bit10~bit0.

Stack

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

If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but acknowledge signal will be inhibited. When the stack pointer is decremented (by RET or RETI), the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. In a similar case, if the stack is full and a "CALL" is subsequently executed, stack overflow occurs and the first entry will be lost (only the most recent four return addresses is stored).

Arithmetic and Logic Unit – ALU

This circuit performs 8-bit arithmetic and logic operation. The ALU provides the following functions:

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

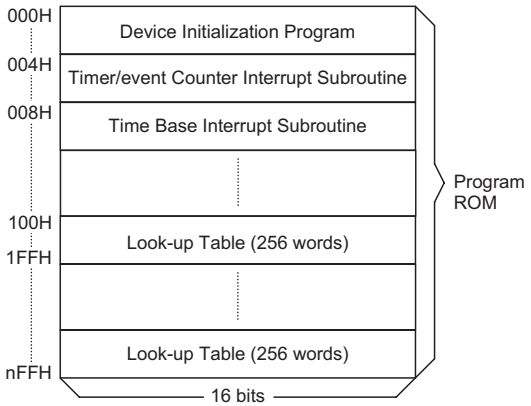
The ALU not only saves the results of a data operation but can also change the status register.

Program Memory – ROM

The program memory is used to store the program instructions, which are to be executed. It also contains data, table information and interrupt entries, and is organized into 1024×16 bits for HT47C07L or 2048×16 bits for HT47C08L, addressed by the program counter and table pointer registers.

Certain locations within the program memory are reserved for special usage:

- Location 000H
  This area is reserved for use by the chip reset for program initialization. After a chip reset is initiated, the program will jump to this location and begin execution.
- Location 004H
  This area is reserved for the timer/event counter interrupt service program. If timer interrupt results from a timer/event counter A or B overflow, and if the interrupt is enabled and the stack is not full, the program will jump to this location and begin execution.
- Location 008H
  This area is reserved for the time base interrupt service program. If a time base interrupt occurs, and if the interrupt is enabled and the stack is not full, the program will jump to this location and begin execution.
- Table location
  Any location within the program memory can be used as a look-up table where programmers can store fixed data. 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 higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined, the higher-order byte of the table word are transferred to the TBLH. The table higher-order byte (TBLH) is a read only register. The table pointer (TBLP) is a read/write register (07H), which indicates the table location. Before accessing the table, the location must be placed in the TBLP. The TBLH is read only and cannot be restored. If the main routine and the ISR (interrupt service routine) both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR. Errors can occur. In other words using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt is supposed to be disabled prior to the table read instruction. It will not be enabled until the TBLH has been backed up. All table related instructions need two cycles to complete the operation. These areas may function as normal program memory depending upon the requirements.



Note: n is up to 3 for HT47C07L while n is up to 7 for HT47C08L

Program Memory

Instruction(s)	Table Location										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: *10~*0: Bits of table location @7~@0: Bits of table pointer

P10~P0: Bits of current program counter

For HT47C07L, the bits of table location are 10 bits.

For HT47C08L, the bits of table location are 11 bits.

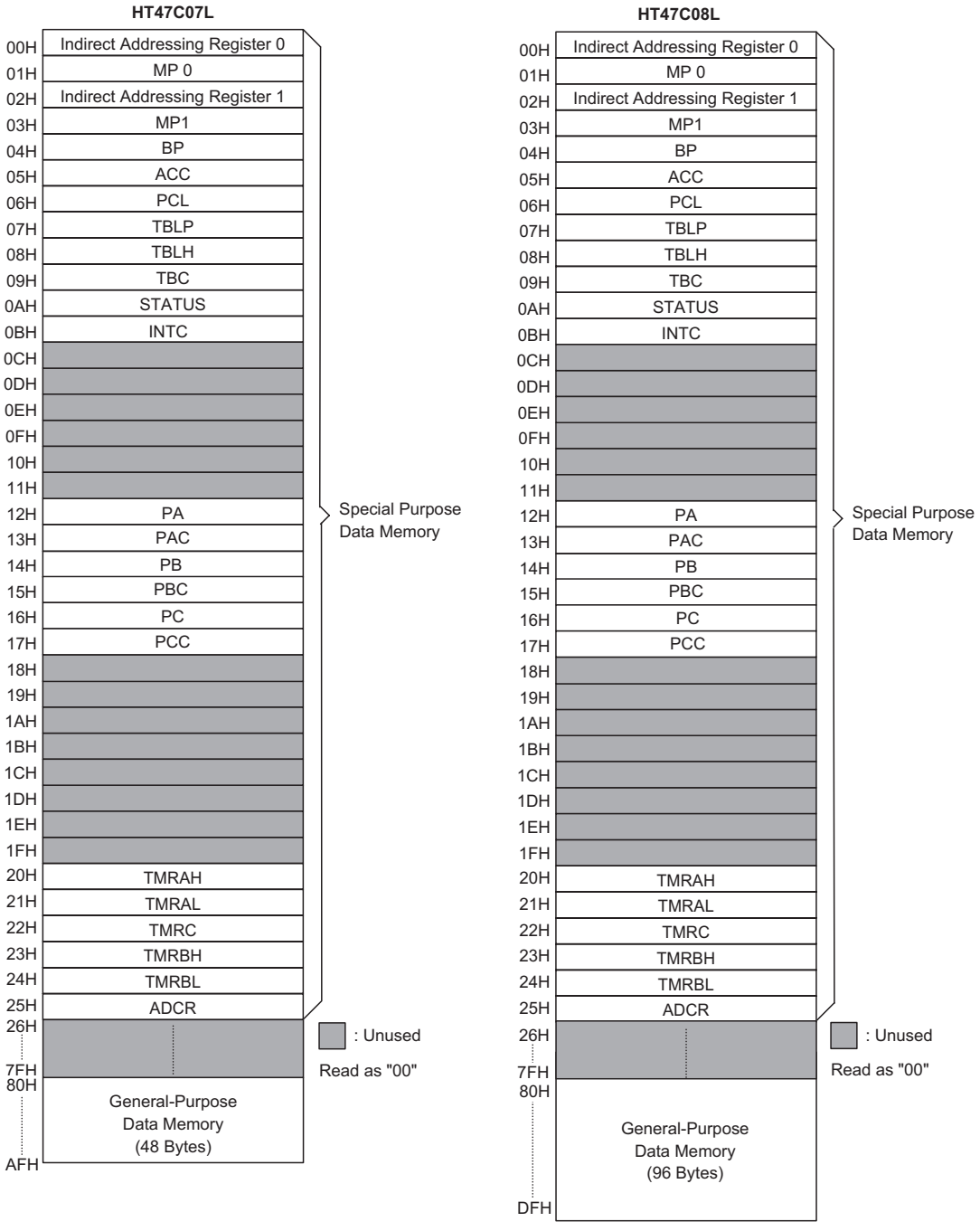
Data Memory – RAM

The data memory is divided into two functional groups: special function registers and general-purpose data memory. Most are read/write, but some are read only.

The special function registers include the indirect addressing register 0 (IAR0), the memory pointer register 0 (MP0), the indirect addressing register 1 (IAR1), the memory pointer register 1 (MP1), the bank pointer (BP), the accumulator (ACC), the program counter lower-order byte register (PCL), the table pointer (TBLP), the table higher-order byte register (TBLH), the time base control register (TBC), the status register (STATUS), the interrupt control register (INTC), the I/O registers (PA; PB; PC), I/O port control register (PAC; PBC; PCC), the timer/event counter A higher-order byte register (TMRAH), the timer/event counter A lower-order byte register (TMRAL), the timer/event counter control register (TMRC), the timer/event counter B higher-order byte register (TMRBH), the timer/event counter B lower-order byte register (TMRBL), the RC oscillator type A/D converter control register (ADCR).

The remaining space before the 80H are reserved for future expanded usage and reading these locations will return the result 00H. The general-purpose data memory, addressed from 80H to AFH for HT47C07L or from 80H to DFH for HT47C08L, is used for data and control information under instruction command.

All data memory areas can handle arithmetic, logic, increment, decrement and rotate operations. Except for some dedicated bits, each bit in the data memory can be set and reset by the "SET [m].i" and "CLR [m].i" instruction, respectively. They are also indirectly accessible through memory pointer registers (MP0;01H, MP1;03H).


HT47C07L
00H Indirect Addressing Register 0
01H MP 0
02H Indirect Addressing Register 1
03H MP1
04H BP
05H ACC
06H PCL
07H TBLP
08H TBLH
09H TBC
0AH STATUS
0BH INTC
0CH
0DH
0EH
0FH
10H
11H
12H PA
13H PAC
14H PB
15H PBC
16H PC
17H PCC
18H
19H
1AH
1BH
1CH
1DH
1EH
1FH
20H TMRAH
21H TMRAL
22H TMRC
23H TMRBH
24H TMRBL
25H ADCR
26H
7FH
80H
AFH
General-Purpose Data Memory (48 Bytes)
Special Purpose Data Memory
: Unused
Read as "00"
HT47C08L
00H Indirect Addressing Register 0
01H MP 0
02H Indirect Addressing Register 1
03H MP1
04H BP
05H ACC
06H PCL
07H TBLP
08H TBLH
09H TBC
0AH STATUS
0BH INTC
0CH
0DH
0EH
0FH
10H
11H
12H PA
13H PAC
14H PB
15H PBC
16H PC
17H PCC
18H
19H
1AH
1BH
1CH
1DH
1EH
1FH
20H TMRAH
21H TMRAL
22H TMRC
23H TMRBH
24H TMRBL
25H ADCR
26H
7FH
80H
DFH
General-Purpose Data Memory (96 Bytes)
Special Purpose Data Memory
: Unused
Read as "00"

RAM Mapping

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

Location 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation to [00H] and [02H] access data memory pointed to by MP0 (01H) and MP1 (03H) respectively. Reading location 00H or 02H indirectly will return a result of 00H. Writing indirectly results in no operation.

The function of data movement between two indirect addressing registers are not supported. The memory pointer registers, MP0 and MP1, are both 8-bit registers which can be used to access the data memory by combining corresponding indirect addressing registers.

MP0 only can be applied to data memory, while MP1 can be applied to the data memory and the LCD display memory.

Accumulator – ACC

The accumulator is closely related with operations carried out by the ALU. It is mapped to location 05H of the data memory and is the place where all immediate results from the ALU are stored. Data movement between two data memory locations must pass through the accumulator.

Status Register – STATUS

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

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flags. In addition it should be noted that operations related to the status register may give different results from those intended. The TO and PDF flags can only be changed by the watchdog timer overflow, system power-up, clearing the watchdog timer and executing the HALT instruction.

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

- **C** is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- **AC** is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- **Z** is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- **OV** is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- **PDF** is cleared by a system power-up or executing the “CLR WDT” instruction. PDF is set by executing the “HALT” instruction.
- **TO** is cleared by a system power-up or executing the “CLR WDT” or “HALT” instruction. TO is set by a WDT time-out.

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

STATUS Register

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

"x" unknown

Bit 7, 6 Unimplemented, read as "0"

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

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

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

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

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

Bit 0 **C:** Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

C is also affected by a rotate through carry instruction.

Oscillator Configuration

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a frequency of 32kHz, 64kHz or 128kHz determined by the mask options. The oscillation frequency may vary with the power supply voltage, temperature and process variations.

Watchdog Timer

The WDT clock source (f_S) is f_{SYS}. The timer is designed to prevent software malfunctions or sequences from jumping to unknown locations with unpredictable results. The Watchdog Timer can be disabled by mask option. If the Watchdog Timer is disabled, all the executions related to the WDT result in no operation.

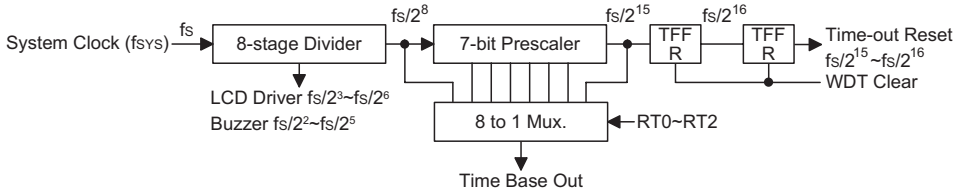
If the Watchdog Timer function is enabled, and the devices enter the power down mode by executing the HALT instruction, the Watchdog Timer will keep counting and then wake up the devices from the power down mode due to the Watchdog Timer time-out as the oscillator is selected to be switched on in the power down mode.

The WDT overflow under normal operation will initialize a "chip reset" and set the status bit TO. Whereas in the power down mode, the overflow will initialize a "warm reset" wherein only the program counter and stack pointer are reset to zero. To clear the contents of the WDT, three methods are adopted. The first is an external hardware reset (a low level on the $\overline{RES}$ pin), the second is via software instructions, and the third is via a "HALT" instruction. The software instruction is CLR WDT. Any execution of the CLR WDT instruction will clear the WDT. The WDT may reset the chip due to time-out.

The WDT time-out period ranges from $f_S/2^{15}$~$f_S/2^{16}$. The "CLR WDT" instruction only clears the last two-stage of the WDT.

Multi-function Timer

The devices provide a multi-function timer for the WDT and time base but with different time-out periods. The multi-function timer consists of an 8-stage divider and a 7-bit prescaler, with the clock source coming from f_{SYS}. The multi-function timer also provides a selectable frequency signal (ranges from $f_S/2^3$ to $f_S/2^6$) for LCD driver circuits, and a selectable frequency signal (ranges from $f_S/2^2$ to $f_S/2^5$) for the buzzer output by options. It is recommended to select a near 4kHz signal to LCD driver circuits for proper display.



Multi-function Timer

Time Base

The time base is used to supply a regular internal interrupt. Its time-out frequency ranges from $f_S/2^8$ to $f_S/2^{15}$ by software programming. Writing data to RT2, RT1 and RT0 (bits 2, 1, 0 of TBC;09H) yields various time-out frequency. If a time base time-out occurs, the related interrupt request flag (TBF; bit 5 of INTC) is set. But if the interrupt is enabled, and the stack is not full, a subroutine call to location 08H occurs. When the HALT instruction is executed, the time base still works and can wake-up from HALT mode if f_{OSC} is on. If the TBF is set to "1" before entering the HALT mode, the wake-up function will be disabled.

TBC Register

Register Name	Bit							
	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	RT2	RT1	RT0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	1	1	1

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **RT2~RT0:** Time Base Divided Factor

000: 2^8
001: 2^9
010: 2^{10}
011: 2^{11}
100: 2^{12}
101: 2^{13}
110: 2^{14}
111: 2^{15}

Reset and Initialisation

There are three ways in which a reset may occur.

- $\overline{RES}$ reset during normal operation
- $\overline{RES}$ reset during power down mode
- WDT time-out reset during normal operation

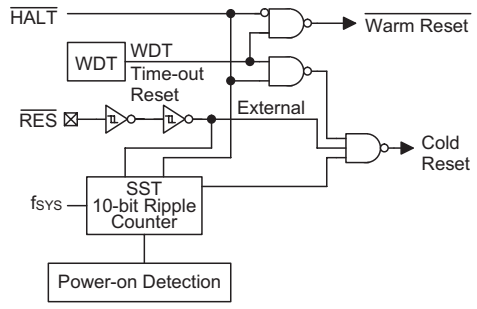
The WDT time-out reset during power down mode is different from other chip reset conditions, since it can perform a warm reset that just resets the program counter and stack pointer leaving the other circuits in their original state. Some registers remain unchanged during other reset conditions. Most registers are reset to the initial condition when the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between different "chip resets".

TO	PDF	RESET Conditions
0	0	System power-up
u	u	$\overline{RES}$ reset during normal operation
0	1	$\overline{RES}$ reset wake-up from power down mode
1	u	WDT time-out during normal operation
1	1	WDT wake-up from power down mode

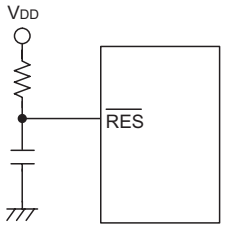
Note: "u" stands for unchanged

The following table indicates the way in which the various functional units are affected after a reset occurs.

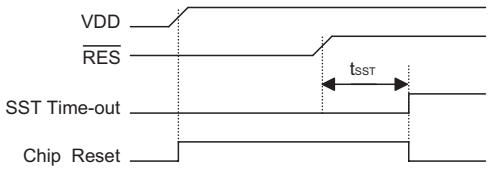
Item	Condition After Reset
Program Counter	Reset to 000H
Interrupts	All interrupts will be disabled
Prescaler, Divider	All timer counter prescaler, divider will be cleared
WDT, Time Base	Clear after master reset, WDT begins counting
Timer/event Counter	All timer counters will be turned off
Input/output Ports	All I/O ports will be setup as inputs
Stack Pointer	Stack pointer will point to the top of the stack



Reset Configuration



Reset Circuit



Reset Timing Chart

The states of the registers are summarized in the following table:

HT47C07L

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (HALT)	WDT Time-out (HALT)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	---- ---0	---- ---0	---- ---0	---- ---0	---- ---0
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBC	---- -111	---- -111	---- -111	---- -111	---- -uuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	---- --11	---- --11	---- --11	---- --11	---- --uu
PCC	---- --11	---- --11	---- --11	---- --11	---- --uu
TMRAH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRAL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	-000 1---	-000 1---	-000 1---	-000 1---	-uuu u---
TMRBH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRBL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	-00x 0000	-00x 0000	-00x 0000	-00x 0000	-uuu uuuu

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

HT47C08L

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	---- ---0	---- ---0	---- ---0	---- ---0	---- ---0
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBC	---- -111	---- -111	---- -111	---- -111	---- -uuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	---1 1111	---1 1111	---1 1111	---1 1111	---u uuuu
PCC	---1 1111	---1 1111	---1 1111	---1 1111	---u uuuu
TMRAH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRAL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	-000 1---	-000 1---	-000 1---	-000 1---	-uuu u---
TMRBH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRBL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	000x 0000	000x 0000	000x 0000	000x 0000	uuuu uuuu

Note: “u” stands for unchanged
“x” stands for “unknown”
“-” stands for unimplemented

Input/Output Ports

There are bidirectional input/output ports in the microcontroller, labeled PA, PB and PC, which are mapped to the data memory [12H], [14H] and [16H] respectively. All of these I/O lines can be used as input and output operations. For the input operation, these lines are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction "MOV A, [m]" (m=12H, 14H or 16H). For output operation, all the data is latched and remain unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC and PCC) to control the input/output configuration. With this control register, CMOS output or Schmitt trigger input with pull-high resistor structures can be reconfigured dynamically under software control. To function as an input, the corresponding latch of the control register has to be set as "1". The pull-high resistor will be exhibited automatically. The input sources also depend on the control register. If the control register bit is "1", the input will read the pad state ("MOV" and read-modify-write instructions). If the control register bit is "0", the contents of the latches will move to internal data bus ("MOV" and read-modify-write instructions). The input paths (pad state or latches) of read-modify-write instructions are dependent on the control register bits. For output function, CMOS is the only configuration. The control registers are mapped to locations 13H, 15H and 17H respectively.

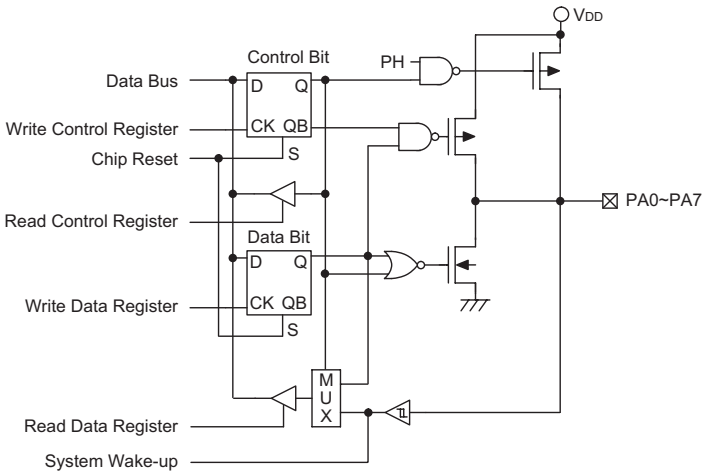
After a chip reset, these input/output lines remain at high levels (pull-high), except PC0/PC1 as floating. Each bit of these input/output latches can be set or cleared by "SET [m].i" (m=12H, 14H or 16H) instructions. Some instructions first input data and then follow the output operations. For example, "SET [m].i", "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 bit of the port A has the capability of waking-up the devices.

The PA0 and PA1 are pin-shared with BZ and $\overline{BZ}$, respectively. If the BZ mode is selected, the output signal in output mode of PA0 (or PA1) will be BZ (or $\overline{BZ}$) signal. The input mode always retain its original functions. The 4kHz buzzer output signals (in output mode) are controlled by the PA0 and PA1 data registers. The truth table of PA0/BZ and PA1/$\overline{BZ}$ are listed below.

PA1 Data Register	PA0 Data Register	PA1, PA0 Pad Function
0 (CLR PA.1)	0 (CLR PA.0)	PA0=BZ, PA1=$\overline{BZ}$
1 (SET PA.1)	0 (CLR PA.0)	PA0=BZ, PA1=0
X	1 (SET PA.0)	PA0=0, PA1=0

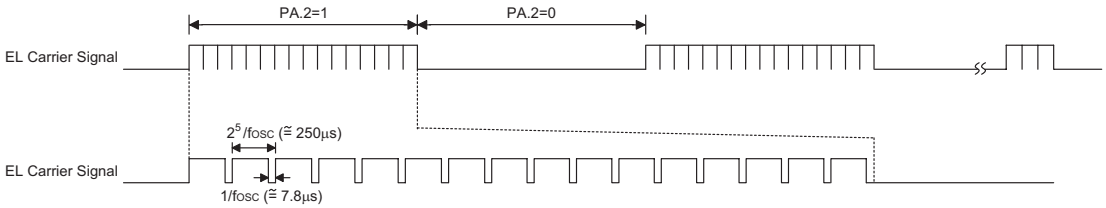
Mask Option



Note: BZ and EL mode functions are not shown in this diagram

The PA2 is pin-shared with EL carrier signals. If the EL carrier output is selected, the output signal in output mode of PA2 will be the EL carrier signal. The input mode always remains its original functions. The EL carrier output signal (in output mode) is controlled by the PA2 data register. The truth table of PA2/EL is listed below.

PA2 Data Register	PA2 Pad Function
0 (CLR PA.2)	PA2=0
1 (SET PA.2)	PA2=EL carrier output



EL Timing (f_{osc}=128kHz)

I/O Register Lists

- **HT47C07L**

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	D7	D6	D5	D4	D3	D2	D1	D0
PAC	D7	D6	D5	D4	D3	D2	D1	D0
PB	D7	D6	D5	D4	D3	D2	D1	D0
PBC	D7	D6	D5	D4	D3	D2	D1	D0
PC	—	—	—	—	—	—	D1	D0
PCC	—	—	—	—	—	—	D1	D0

- **HT47C08L**

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	D7	D6	D5	D4	D3	D2	D1	D0
PAC	D7	D6	D5	D4	D3	D2	D1	D0
PB	D7	D6	D5	D4	D3	D2	D1	D0
PBC	D7	D6	D5	D4	D3	D2	D1	D0
PC	—	—	—	D4	D3	D2	D1	D0
PCC	—	—	—	D4	D3	D2	D1	D0

PAC Register

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

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

PBC Register

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

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

PCC Register

- **HT47C07L**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 I/O Port C bit 1~bit 0 Input/Output Control
0: Output
1: Input

- **HT47C08L**

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

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 I/O Port C bit 4~bit 0 Input/Output Control
0: Output
1: Input

Timer/Event Counter

One 16-bit timer/event counter or RC type A/D converter is implemented in the devices. The ADC_$\overline{TM}$ bit (bit 1 of ADCR register) determines whether timer A and timer B are composed of one 16-bit timer/event counter or composed of an RC type A/D converter.

The TMRAL, TMRAH, TMRBL and TMRBH composed of one 16-bit timer/event counter, when ADC_$\overline{TM}$ bit is "0". The TMRBL and TMRBH are timer/event counter preload registers for lower-order byte and higher-order byte respectively.

The timer/event counter clock source comes from system clock (f_{SYS}) or external source. The external clock input allows the user to count external events, count external RC type A/D clock, measure time intervals or pulse widths, or generate an accurate time base.

There are six registers related to the timer/event counter operating mode. TMRAH ([20H]), TMRAL ([21H]), TMRC ([22H]), TMRBH ([23H]), TMRBL ([24H]) and ADCR ([25H]). Writing to TMRBL only writes the data into a low byte buffer, and writing to TMRBH will write the data and the contents of the low byte buffer into the time/event counter preload register (16-bit) simultaneously. The timer/event counter preload register is changed by writing to TMRBH operations and writing to TMRBL will keep the timer/event counter preload register unchanged.

Reading TMRAH will also latch the TMRAL into the low byte buffer to avoid false timing problem. Reading TMRAL returns the contents of the low byte buffer. In other words, the low byte of the timer/event counter can not be read directly. It must read the TMRAH first to make the low byte contents of timer/event counter be latched into the buffer.

The TMRC is the timer/event counter control register, which defines the timer/event counter options. The timer/event counter control register defines the operating mode, counting enable or disable and active edge. Writing to timer B location puts the starting value in the timer/event counter preload register, while reading timer A yields the contents of the timer/event counter. Timer B is the timer/event counter preload register.

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 (A/D clock) comes from an external pin. The timer mode functions as a normal timer with the clock source coming from the internal clock (f_{SYS}). Finally, the pulse width measurement mode can be used to count the high or low level duration of the external signal (A/D clock from pad: RCINn) (HT47C07L: RCIN; HT47C08L: RCIN0 or RCIN1 selected by CHSEL). The counting is based on the system clock (f_{SYS}).

In the event count, A/D clock or internal timer mode, once the timer/event counter starts counting, it will count from the current contents in the timer/event counter (TMRAH and TMRAL) to FFFFH. Once overflow occurs, the counter is reloaded from the timer/event counter preload register (TMRBH and TMRBL) and at the same time generates the corresponding interrupt request flag (TF; bit 4 of INTC).

In the pulse width measurement mode with the TON and TE bits equal to one, once the RCIN has received a transient from low to high (or high to low if the TE bit is 0) it will start counting until the A/D Clock returns to the original level and resets the TON. The measured result will remain in the timer/event counter even if the activated transient occurs again. In other words, only one cycle measurement can be done. Until setting the TON, the cycle measurement will function again as long as it receives further transient pulse. Note that in this operation mode, the timer/event counter starts counting not according to the logic level but according to the transient edges. In the case of counter overflow, the counter is reloaded from the timer/event counter preload register and issues interrupt request just like the other two modes.

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 will automatically be cleared after the measurement cycle is completed. But in the other two modes, the TON can only be reset by instructions.

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 turns on, data written to the timer/event counter preload register is kept only in the timer/event counter preload register. The timer/event counter will still operate until overflow occurs.

When the timer/event counter (reading TMRAH) is read, the clock will be blocked to avoid errors. As this may results in a counting error, this must be taken into consideration.

It is strongly recommended to load first the desired value into TMRBL, TMRBH, TMRAL, and TMRAH registers then turn on the related timer/event counter for proper operation. Because the initial value of TMRBL, TMRBH, TMRAL and TMRAH are unknown.

Example for Timer/event counter mode (disable interrupt):

```
clr tmrc
clr adcr.1                  ; set timer mode
clr intc.4                  ; clear timer/event counter interrupt request flag
mov a, low (65536-1000)     ; give timer initial value
mov tmrbl, a                ; count 1000 time and then overflow
mov a, high (65536-1000)
mov tmrbh, a
mov a, 01010000b            ; timer clock source=fSYS and timer on
mov tmrc, a
p10:
clr wdt
```

TMRC Register

Register Name	Bit							
	7	6	5	4	3	2	1	0
Name	—	TM1	TM0	TON	TE	—	—	—
R/W	—	R/W	R/W	R/W	R/W	—	—	—
POR	—	0	0	0	1	—	—	—

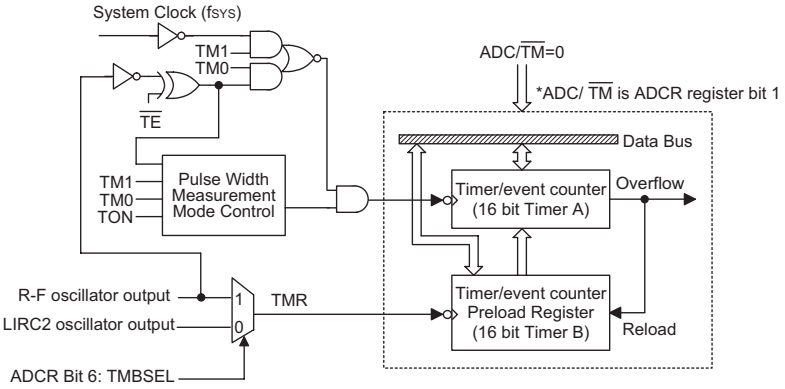
Bit 7 Unimplemented, read as "0"

Bit 6~5 **TM1, TM0:** TO define the operating mode
00: Unused
01: Event counter mode (External clock: A/D clock from pad RCIN)
10: Timer mode (Internal clock: f_{SYS})
11: Pulse width measurement mode (RCIN, f_{SYS})

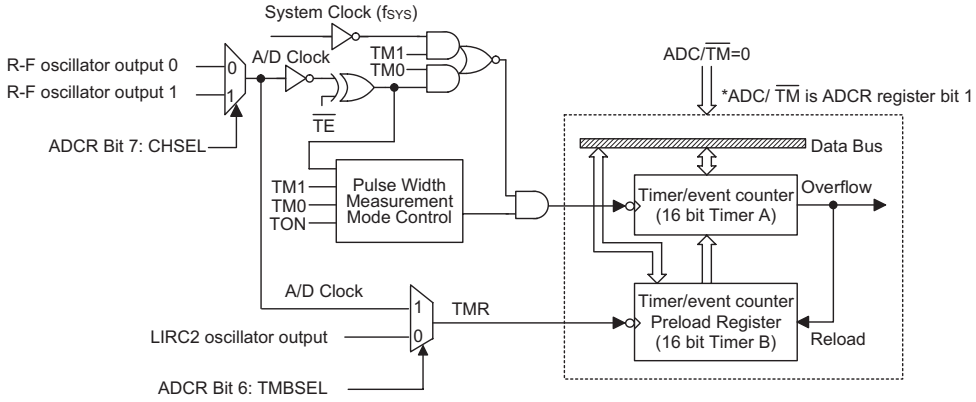
Bit 4 **TON:** To enable/disable timer counting
0: Disabled
1: Enabled

Bit 3 **TE:** Defines the TMR active edge of the timer/event counter
In Event Counter Mode (TM1,TM0)=(0,1)
1: count on falling edge
0: count on rising edge
In Pulse Width measurement mode (TM1,TM0)=(1,1)
1: start counting on the rising edge, stop on the falling edge
0: start counting on the falling edge, stop on the rising edge

Bit 2~0 Unimplemented, read as "0"



Timer/Event Counter – HT47C07L



Timer/Event Counter – HT47C08L

RC Type A/D Converter

An RC type A/D converter is implemented in the devices. The A/D converter contains two 16-bit programmable count-up counters and the timer A clock source comes from the system clock. The timer B clock source comes from the external RC oscillator or internal RC oscillator selected by the TMBSEL bit in the ADCR register. The TMRAL, TMRAH, TMRBL and TMRBH are composed of the A/D converter when ADC_$\overline{TM}$ bit (bit 1 of ADCR register) is to 1.

The A/D converter timer B clock source may come from R-F Oscillator or the internal RC oscillator for HT47C07L device. For HT47C08L device, the A/D converter timer B clock source may come from R-F Oscillator 0, R-F Oscillator 1 or the internal RC oscillator where n is equal to 0 or 1. The timer A clock source is the system clock by setting (TM1, TM0=1, 0).

There are six registers related to the A/D converter, i.e., TMRAH, TMRAL, TMRC, TMRBH, TMRBL and ADCR. The internal timer clock is input to TMRAH and TMRAL, the A/D clock is input to TMRBH and TMRBL. The bit 0 of the ADCR register, OVB_$\overline{OVA}$, determines whether timer A or timer B overflows, then the TF bit is set and timer interrupt occurs. When the A/D converter mode timer A or timer B overflows, the TON bit is reset and stop counting. Writing TMRAH/TMRBH makes the starting value be placed in the timer A or timer B and reading TMRAH/TMRBH retrieves the contents of the timer A or timer B. Writing TMRAL/TMRBL only writes the data into a low byte buffer, and writing TMRAH/TMRBH will write the data and the contents of the low byte buffer into the timer A or timer B (16-bit) simultaneously. The timer A or timer B is changed by writing TMRAH/TMRBH operations and writing TMRAL/TMRBL will keep the timer A or timer B unchanged.

Reading TMRAH/TMRBH will also latch the TMRAL/TMRBL into the low byte buffer to avoid false timing problem. Reading TMRAL/TMRBL returns the contents of the low byte buffer. In other words, the low byte of timer A or timer B cannot be read directly. It must read the TMRAH/TMRBH first to make the low byte contents of timer A or timer B be latched into the buffer.

The bit 2 of ADCR decides which resistor and capacitor compose an oscillation circuit and input to TMRBH and TMRBL.

The TM0 and TM1 bits of TMRC define the timer A clock source. It is recommended that the timer A clock source use the system clock.

When the TON bit (bit 4 of the TMRC) is set to "1" the timer A and timer B will start counting until timer A or timer B overflows, the timer/event counter generates the interrupt request flag (TF; bit 4 of INTC) and the timer A and timer B stop counting and reset the TON bit to "0" at the same time.

If the TON bit is "1", the TMRAH, TMRAL, TMRBH and TMRBL cannot be read or written to. Only when the timer/event counter is off and when the instruction "MOV" is used can those four registers be read or written to.

“System Clock Oscillator” is used to oscillating the clock signal to be the MCU system clock. It comsumes fewer power than other oscillators. The system clock is also the reference clock to count the real time in order to decreasing power consumption. However, the system clock is not accurate in variant operating voltage or in variant ambient temperature. Hence, the system clock shall be calibrated first. LIRC2 Oscillator is more accurate than System Clock Oscillator in variant operating voltage or in variant ambient temperature, so LIRC2 can be used to calibrate the System Clock Oscillator. The Calibration can be acheieved by executing the steps below.

- Step1: Select “System clock” to the clock source of TMRA and select “LIRC2” to the clock source of TMRB
- Step2: Assign the values of registers TMRAL and TMRAH
- Step3: Select TMRA as the interrupt source of timer/event overflow occurance and then activate TMRA in timer mode and TMRB simultaneously.
- Step4: Once TMRA overflow occur, stop counting TMRA and TMRB automatically and then calculate the variance of TMRA and TMRB
- Step5: System clock can be calibrated by the variance just calculated in the application program.

In order to decrase power consumption, LIRC2 is advised to be turnrd on while doing System Clock pulse width measurement and turned off while finishing System Clock pulse width measurement.

“R to F Oscillator” is used to oscillate the clock whose frequency is decided by the external resistor. Take the reference to the Bolck diagram “R to F Oscillator”, a reference resistor “REF Resistor” is supplied to “R to F Oscillator” to produce an output clock frequency as a reference frequency. TheThermistor “NIC” is supplied to “R to F Oscillator” to produce an output clock frequency and then compare that frequency with the referency frequency to measure the temperature. The temperature measurement can be achieved by following steps.

- Step1: Select “System clock” to the clock source of TMRA and select “R to F Oscillator” to the clock source of TMRB
- Step2: Select the RREF path as the input RC path of “R to F Oscillator” by clearing RSEL control bit
- Step3: Assign the values of registers TMRAL and TMRAH
- Step4: Select TMRA as the interrupt source of timer/event overflow occurance and then activate TMRA in timer mode and TMRB

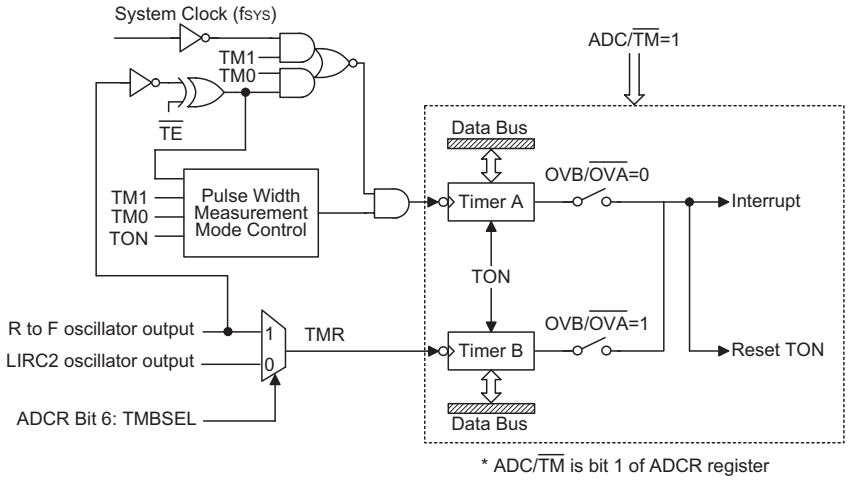
- Step5: Once TMRA overflow occur, stop counting TMRA and TMRB automatically and then calculate the variance of TMRA and TMRB. The variance is said to be "Variance 1"
- Step6: Select the RSEN path as the input RC path of "R to F Oscillator" by setting RSEL control bit and then redo the actions from step 4 to step 5. Furthermore, the variance can be obtained and said to be "Variance 2"
- Step7: The output frequency difference of "RSEN path" and "RREF path" in "R to F Oscillator" can be calculated by comparison of "Variance 1" and "Variance 2" in application program. Furthermore, the NIC temperature can be obtained from that frequency difference just calculated.

R-F Oscillator 0 on/off control for HT47C07L

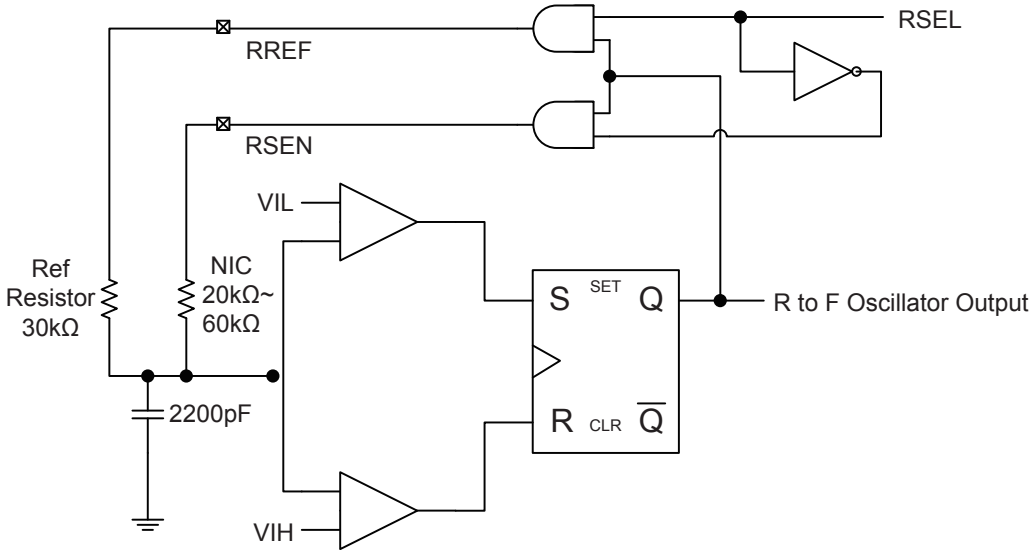
	TM1	TM0	ADC/$\overline{TM}$	TON	R-F OSC0	Notes
TON Off	Don't Care	Don't Care	Don't Care	0	OFF	—
Timer Mode	1	0	0	1	OFF	—
Event Counter Mode	0	1	1	1	ON	The TE bit must be set high immediately after the TON bit has been set high
PWM Mode	1	1	1	1	ON	—
UNUSED	0	0	Don't Care	1	OFF	—
Event Counter Mode or PWM Mode	Don't Care	1	0	1	ON	—

R-F Oscillator 0 and R-F Oscillator 1 on/off control for HT47C08L

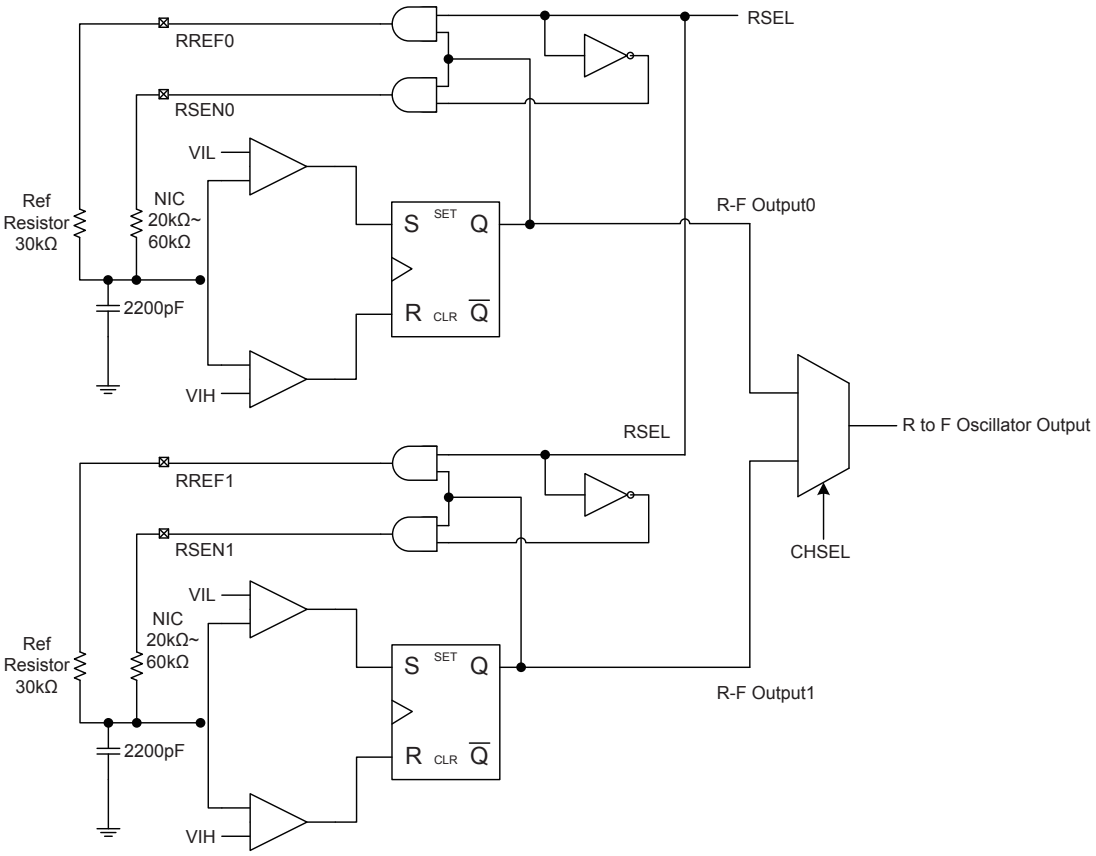
	TM1	TM0	ADC/$\overline{TM}$	CHSEL	TON	R-F OSC0	R-F OSC1	Notes
TON Off	Don't Care	Don't Care	Don't Care	Don't Care	0	OFF	OFF	—
Timer Mode	1	0	0	Don't Care	1	OFF	OFF	—
Event Counter Mode	0	1	1	0	1	ON	OFF	The TE bit must be set high immediately after the TON bit has been set high
Event Counter Mode	0	1	1	1	1	OFF	ON	The TE bit must be set high immediately after the TON bit has been set high
PWM Mode	1	1	1	1	1	OFF	ON	—
PWM Mode	1	1	1	0	1	ON	OFF	—
UNUSED	0	0	Don't Care	Don't Care	1	OFF	OFF	—
Event Counter Mode or PWM Mode	Don't Care	1	0	0	1	ON	OFF	—
Event Counter Mode or PWM Mode	Don't Care	1	0	1	1	OFF	ON	—



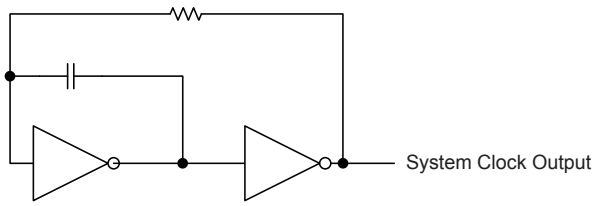
RC Type A/D Converter



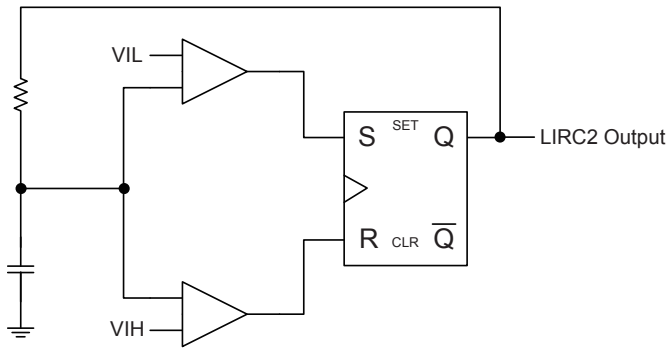
R to F Oscillator - HT47C07L



R to F Oscillator - HT47C08L



System Clock Oscillator



LIRC2 Oscillator

ADCR Register

- **ADCR Register – HT47C07L**

Bit	7	6	5	4	3	2	1	0
Name	—	TMBSEL	LIRC2ON	BLF	BON	RSEL	ADC_$\overline{TM}$	OVB_$\overline{OVA}$
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **TMBSEL:** Timer B clock source select
0: LIRC2 oscillator
1: R to F Oscillator
Note that this bit is only available when the RC type A/D converter is enabled (ADC_$\overline{TM}$=1).

Bit 5 **LIRC2ON:** LIRC 2 oscillator enable control
0: Disable
1: Enable

Bit 4 **BLF:** Low voltage flag
0: Battery power is good
1: Low battery

Bit 3 **BON:** Low voltage detector enable control
0: Disable
1: Enable

Bit 2 **RSEL:** A/D Converter operating mode select
0: RREF~CREF oscillation (reference resistor and reference capacitor)
1: RSEN~CREF oscillation (resistor sensor and reference capacitor)

Bit 1 **ADC_$\overline{TM}$:** RC Type A/D Converter or 16-bit Timer/Event counter enable control
0: 16-bit Timer/Event counter is enabled
1: RC Type A/D Converter is enabled

Bit 0 **OVB_$\overline{OVA}$:** Select timer/event counter interrupt source
0: Timer A overflow
1: Timer B overflow
In the RC type A/D converter mode, this bit is used to define the timer/event counter interrupt which comes from timer A or B overflow. In timer/event counter mode, this bit is not available.

• **ADCR Register – HT47C08L**

Bit	7	6	5	4	3	2	1	0
Name	CHSEL	TMBSEL	LIRC2ON	BLF	BON	RSEL	ADC_$\overline{TM}$	OVB_$\overline{OVA}$
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CHSEL:** Slect the input channel of R-F Oscillator Output
0: R-F Output 0
1: R-F Output 1

Bit 6 **TMBSEL:** Timer B clock source select
0: LIRC2 oscillator
1: R to F oscillator
Note that this bit is only available when the RC type A/D converter is enabled (ADC_$\overline{TM}$=1).

Bit 5 **LIRC2ON:** LIRC 2 oscillator enable control
0: Disable
1: Enable

Bit 4 **BLF:** Low voltage flag
0: Battery power is good
1: Low battery

Bit 3 **BON:** Low voltage detector enable control
0: Disable
1: Enable

Bit 2 **RSEL:** A/D Converter operating mode select
0: RREF~CREF oscillation (reference resistor and reference capacitor)
1: RSEN~CREF oscillation (resistor sensor and reference capacitor)

Bit 1 **ADC_$\overline{TM}$:** RC Type A/D Converter or 16-bit Timer/Event counter enable control
0: 16-bit Timer/Event counter is enabled
1: RC Type A/D Converter is enabled

Bit 0 **OVB_$\overline{OVA}$:** Select timer/event counter interrupt source
0: Timer A overflow
1: Timer B overflow
In the RC type A/D converter mode, this bit is used to define the timer/event counter interrupt which comes from timer A or B overflow. In timer/event counter mode, this bit is not available.

Example for RC type A/D converter mode (Timer A overflow):

```
clr tmrc
clr adcr.1                  ; set timer mode
clr intc.4                  ; clear timer/event counter interrupt request flag
mov a, low (65536-1000)     ; give timer A initial value
mov tmrbl, a                ; count 1000 time and then overflow
mov a, high (65536-1000)
mov tmrbh, a
mov a, 00000010b            ; RREF~CREF
                            ; set RC type ADC mode
                            ; set Timer A overflow
mov adcr, a
mov a, 00h                  ; give timer B initial value
mov tmrbl, a
mov a, 00h
mov tmrbh, a
mov a, 01010000b            ; timer A clock source=fSYS and timer on
mov tmrc, a
p10:
clr wdt
snz intc.4                  ; polling timer/event counter interrupt request flag
jmp p10
clr intc.4                  ; clear timer/event counter interrupt request flag
                            ; program continue
```

Example for RC type A/D converter mode (Timer B overflow):

```
clr tmrc
clr adcr.1                  ; set timer mode
clr intc.4                  ; clear timer/event counter interrupt request flag
mov a, 00h                  ; give timer A initial value
mov tmrbl, a
mov a, 00h
mov tmrbh, a
mov a, 00000011b            ; RREF~CREF
                            ; set RC type ADC mode
                            ; set Timer B overflow
mov adcr,a
mov a, low (65536-1000)     ; give timer B initial value
mov tmrbl, a                ; count 1000 time and then overflow
mov a, high (65536-1000)
mov tmrbh, a
mov a, 00110000b            ; timer A clock source=fSYS and timer on
mov tmrc, a
p10:
clr wdt
snz intc.4                  ; polling timer/event counter interrupt request flag
jmp p10
clr intc.4                  ; clear timer/event counter interrupt request flag
                            ; program continue
```

Power Down Operation – HALT

The HALT mode is initialized by the "HALT" instruction and results in the following:

- The f_{OSC} and f_{SYS} will still work or stop depending on the oscillator/LCD function on/off in the power down mode option, but T1 will be turned off.
- The contents of the on-chip RAM and registers remain unchanged.
- The WDT will be cleared and resume counting if the oscillator is switched on in the power down mode.
- All I/O ports maintain their original status.
- The PDF flag is set and the TO flag is cleared.
- The LCD driver can be turned off or on depending upon that the LCD function is disabled or enabled in the power down mode.
- The time base will stop or keep running depending upon that the oscillator is switched off or on in the power down mode.

Port A wake-up and internal interrupt wake-up methods can be considered as a continuation of normal execution. Awakening from an I/O port stimulus, the program will resume execution at the next instruction. If awakening from an internal interrupt, two possibilities may occur. If the internal interrupt is disabled or the internal interrupt is enabled but the stack is full, the program will resume execution at the next instruction. If the internal interrupt is enabled and the stack is not full, a regular interrupt response takes place.

If an internal interrupt request flag is set to 1 before entering the power down mode, the wake-up function of the related interrupt will be disabled.

If the wake-up results from an internal interrupt acknowledge signal, the actual interrupt subroutine execution will be delayed by more than one cycle. However, if the wake-up results in the next instruction execution following the HALT instruction, the execution will be performed immediately.

To minimize power consumption, all the I/O pins should be carefully managed before entering the power down mode.

Interrupts

The devices provide an internal timer/event counter interrupt and an internal time base interrupt. The interrupt control register (INTC;0BH) contains the interrupt control bits to set the enable/disable and interrupt request flags.

Once an interrupt subroutine is serviced, all other interrupts will be blocked by clearing the EMI bit. This scheme may prevent any further interrupt nesting. Other interrupt requests may occur during this interval, but only the interrupt request flag is recorded. If another interrupt requires servicing while the program is in the interrupt service routine, the programmer should set the EMI bit and the corresponding bit of the INTC to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the stack pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full.

As an interrupt is serviced, a control transfer occurs by pushing the program counter onto the stack, followed by a branch to a subroutine at specified locations in the program memory. Only the program counter is pushed onto the stack. If the contents of the register and status register (STATUS) is altered by the interrupt service program which corrupts the desired control sequence, the contents must be saved first.

The internal timer/event counter interrupt is initialized by setting the timer/event counter interrupt request flag (TF; bit 4 of INTC), caused by a timer A or timer B overflow. When the interrupt is enabled, and the stack is not full and the TF bit is set, a subroutine call to location 04H will occur. The related interrupt request flag (TF) will be reset and the EMI bit cleared to disable further interrupts.

The time base interrupt is initialized by setting the time base interrupt request flag (TBF; bit 5 of INTC), caused by a regular time base signal. When the interrupt is enabled, and the stack is not full and the TBF bit is set, a subroutine call to location 08H will occur. The related interrupt request flag (TBF) will be reset and the EMI bit cleared to disable further interrupts.

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

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

Interrupt Source	Priority	Vector
Timer/event counter interrupt	1	04H
Time base interrupt	2	08H

INTC Register

Register Name	Bit							
	7	6	5	4	3	2	1	0
Name	—	—	TBF	TF	—	ETBI	ETI	EMI
R/W	—	—	R/W	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TBF:** Time base interrupt request flag
0: Inactive
1: Active

Bit 4 **TF:** Timer/event counter interrupt request flag
0: Inactive
1: Active

Bit 3 Unimplemented, read as "0"

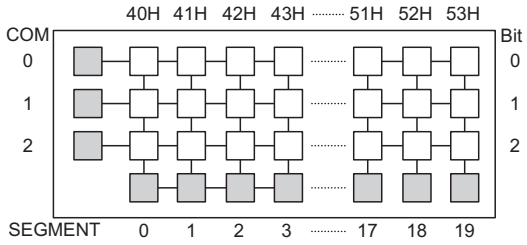
Bit 2 **ETBI:** Controls the time base interrupt
0: Disabled
1: Enabled

Bit 1 **ETI:** Controls the timer/event counter interrupt
0: Disabled
1: Enabled

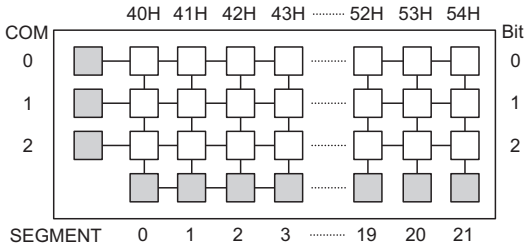
Bit 0 **EMI:** Controls the master or global interrupt
0: Disabled
1: Enabled

LCD Display Memory

The devices provide an area of embedded data memory for LCD display. This area is located in the RAM at Bank 1. 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 01H, any data written into the address from 40H to 53H or from 40H to 54H will affect the LCD display. When the BP is cleared to 00H, any data written into the address from 40H to 53H or from 40H to 54H means to access the general purpose data memory. The LCD display can be read and written to only by indirect addressing mode using MP1. When 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 devices.



LCD Display Memory (Bank 1) – HT47C07L



LCD Display Memory (Bank 1) – HT47C08L

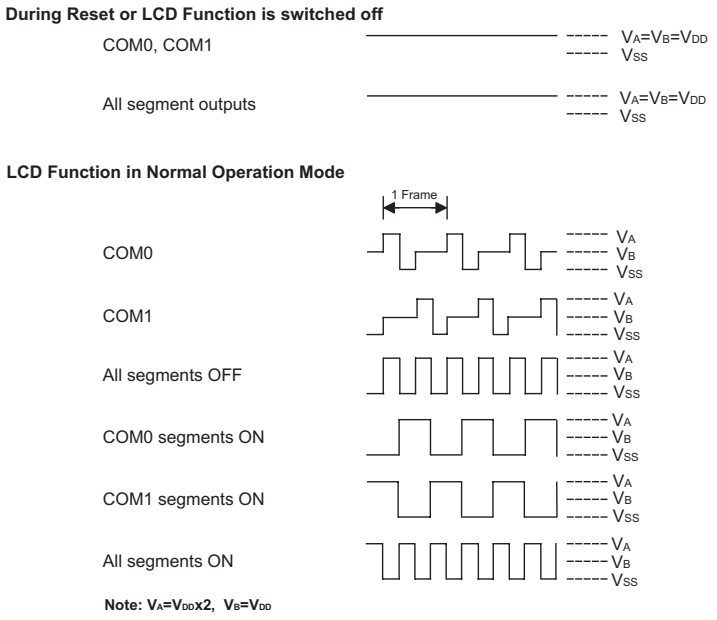
LCD Driver Output

The output number of the LCD driver device can be 20×2 or 19×3 (i.e., 1/2 duty or 1/3 duty) for HT47C07L and 21×3 (i.e., 1/3 duty) for HT47C08L determined by a mask option. The LCD driver bias type can only be “C” type. A capacitor mounted between C1 and C2 pins is needed. A capacitor mounted between VA pin and ground is required if 1/2 bias level is selected. Note that for the HT47C08L device if the 1/3 duty is selected, the LCD driver bias will be set to 1/2 bias. The capacitors with a capacitance value of 0.1μF are suggested to be used for the LCD bias generator. The relationship between LCD bias type, bias levels and capacitor connections are listed in the following table.

Bias Types	Bias Levels	C1/C2	VA Pin	VC Pin
C	1/2	0.1μF	0.1μF	x

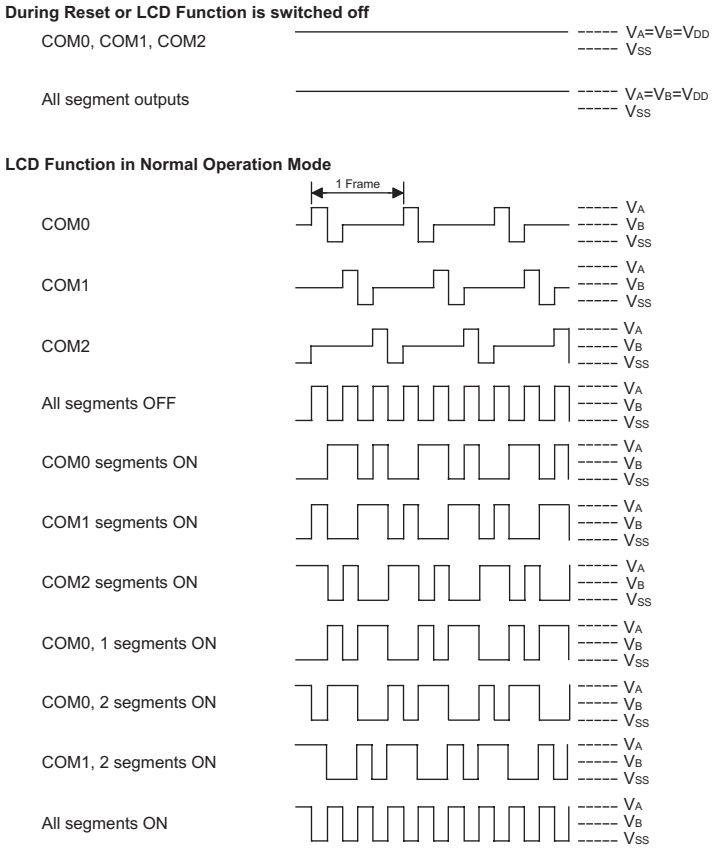
A clock source is necessary to drive the LCD driver and the recommended clock frequency to drive the LCD driver is about 4kHz. Since the LCD driver clock is derived from the system clock, the system clock should be divided by a selected ratio determined by mask options to obtain a proper LCD driver clock frequency.

Clock Source Frequency	LCD Clock Division Ratio	LCD Driver Clock Frequency
128kHz	$1/2^5$	4kHz
64kHz	$1/2^4$	4kHz
32kHz	$1/2^3$	4kHz



Note: $V_A=V_{DD}\times 2$, $V_B=V_{DD}$

LCD Driver Output (1/2 duty, 1/2 bias)



Note: $V_A=V_{DD}\times2$, $V_B=V_{DD}$

LCD Driver Output (1/3 duty, 1/2 bias)

Low Voltage Detector – LVD

The devices provide a low voltage detector for battery system application. If the LVD is on and the battery voltage is lower than the specified value, the low voltage flag (BLF; bit 4 of ADCR register) is set. The specified value may be set as 1.3V±0.05V, 1.27V±0.05V or 1.25V±0.05V by mask options. The low voltage detector circuit can be turned "On" or "Off" by writing a "1" or a "0" to BON (bit 3 of ADCR register). The BLF is invalid when the BON is cleared as "0". Set BON=0 after checking the voltage to prevent from DC current consumption of LVD.

Mask Options

The following table shows many kinds of options in the devices. All these options should be defined in order to ensure proper system functioning.

No.	Options
1	Internal RC oscillator frequency selection: 32kHz, 64kHz or 128kHz
2	LVD voltage selection: 1.25V, 1.27V or 1.3V
3	WDT function: Enable or disable
4	Buzzer output frequency selection: $f_{SYS}/2^2$, $f_{SYS}/2^3$, $f_{SYS}/2^4$ or $f_{SYS}/2^5$
5	To define the PA0/PA1 output function: Normal I/O function or Buzzer output (PA0: BZ; PA1: $\overline{BZ}$)
6	To define the PA2 output function: Normal I/O function or EL carrier output
7	Oscillator/LCD are on or off when CPU enters power down mode: Off or on
8	PA0~PA7 pull-high function: Enable or disable
9	PB0~PB7 pull-high function: Enable or disable
10	PC0~PC1 for HT47C07L/PC0~PC4 for HT47C08L pull-high function: Enable or disable
11	LCD segment selection (SEG9~SEG18): I/O function or LCD segment
12	LCD duty selection(*): For HT47C07L: 1/2 duty (2 common) or 1/3 duty (3 common) For HT47C08L: 1/3 duty (3 common, 1/2 bias)
13	LCD driver clock selection: $f_{SYS}/2^3$, $f_{SYS}/2^4$, $f_{SYS}/2^5$ or $f_{SYS}/2^6$

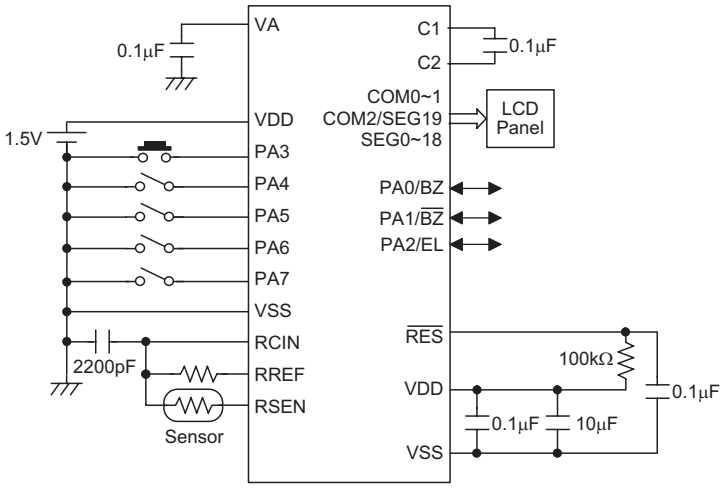
For HT47C07L device:

If the 1/2 duty is selected, the COM2/SEG19 pin is used as a LCD segment output SEG19.

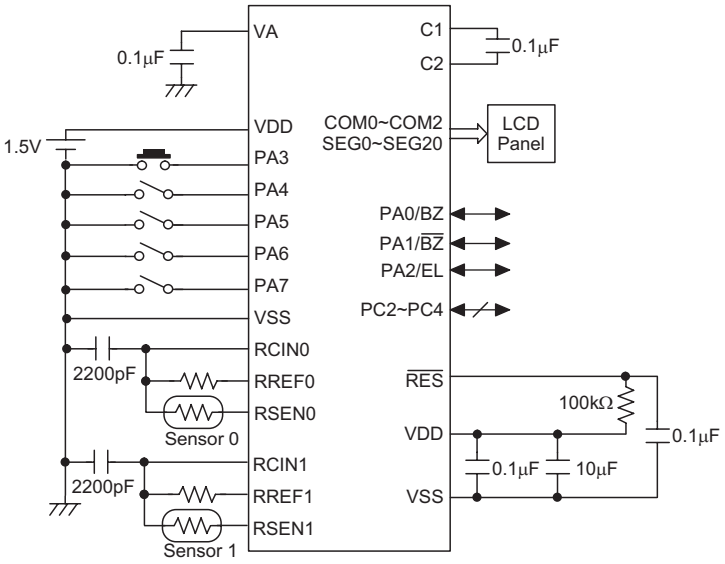
If the 1/3 duty is selected, the COM2/SEG19 pin is used as a LCD common output COM2.

Application Circuits

HT47C07L



HT47C08L



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

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

Mnemonic	Description	Cycles	Flag Affected
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRDC [m]	Read table to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

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

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

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

Instruction Definition

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

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

Operation: ACC ← ACC + [m] + C

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

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

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

Operation: [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation: ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation: ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation: [m] ← ACC + [m]

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

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

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

Operation: ACC ← ACC "AND" [m]

Affected flag(s): Z

AND A,x Logical AND immediate data to ACC

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

Operation: ACC ← ACC "AND" x

Affected flag(s): Z

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

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

Operation: [m] ← ACC "AND" [m]

Affected flag(s): Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

Operation: ACC ← ACC − [m] − C

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

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

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

Operation: [m] ← ACC − [m] − C

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s): None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

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

Operation Skip if [m].i=0

Affected flag(s) None

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

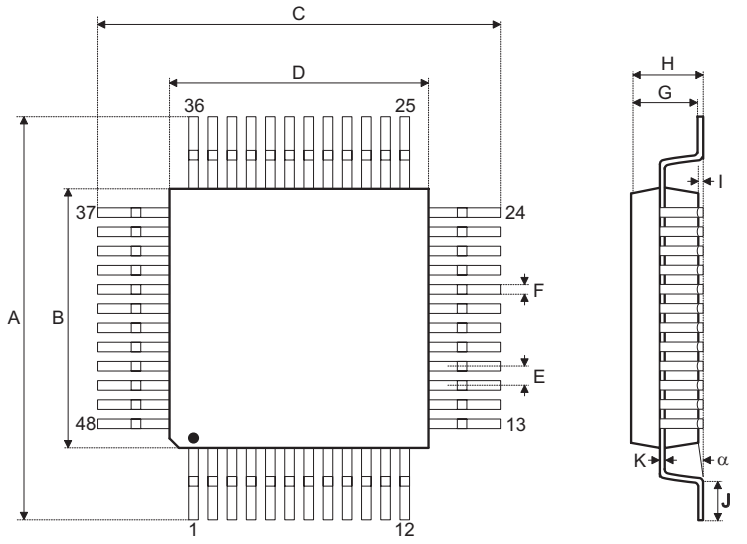
Package Information

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 information.

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

- Further Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- Packing Meterials Information
- Carton information

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



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.020 BSC	—
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.50 BSC	—
F	0.17	0.22	0.27
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°