HT47R20A-1/HT47C20-1
R-F Type 8-Bit MCU

Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0029E Using the Time Base Function in the HT47R20A-1
  - HA0030E Using the RTC in the HT47R20A-1
  - HA0034E Using the Buzzer Function in the HT47R20A-1
  - HA0036E Using the PFD Function in the HT47R20A-1
  - HA0045E Distinguishing between the Different Devices in the HT47 MCU Series

Features

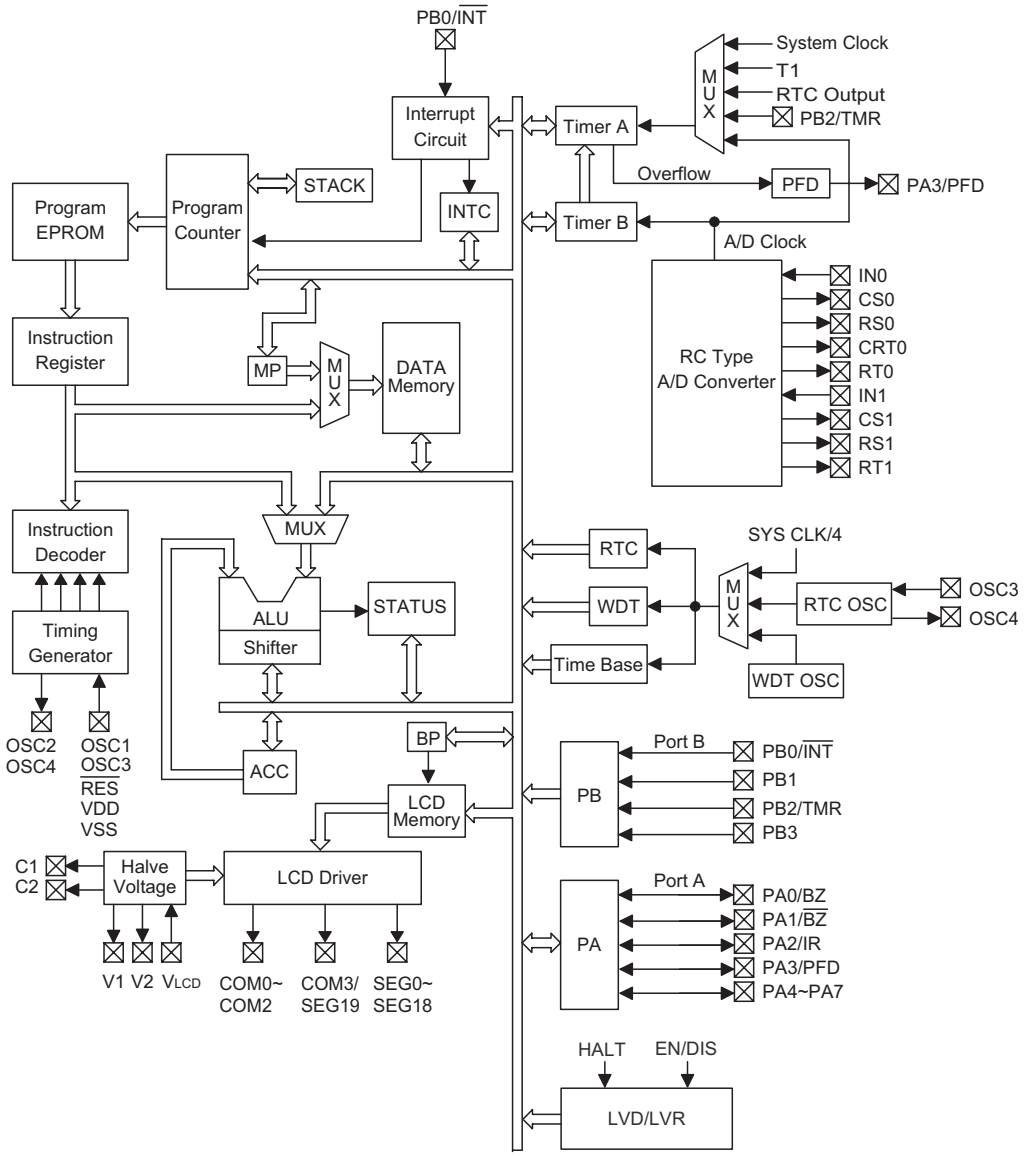
- Operating voltage: 2.2V~5.5V
- Eight bidirectional I/O lines
- Four input lines
- One interrupt input
- One 16-bit programmable timer/event counter with PFD (programmable frequency divider) function
- On-chip crystal and RC oscillator for system clock
- One 32.768kHz crystal oscillator for real time clock or system clock
- Watchdog Timer
- 2K×16 program memory
- 64×8 data memory RAM
- One Real Time Clock (RTC)
- One 8-bit prescaler for RTC
- One low voltage detector
- One low voltage reset circuit
- One buzzer output
- HALT function and wake-up feature reduce power consumption
- LCD bias C type or R type
- One LCD driver with 20×2, 20×3 or 19×4 segments
- One IR carrier output
- Two channels RC type A/D converter
- Four-level subroutine nesting
- Bit manipulation instruction
- 16-bit table read instruction
- Up to 1μs instruction cycle with 4MHz system clock
- All instructions in one or two machine cycles
- 63 powerful instructions
- 64-pin LQFP package

General Description

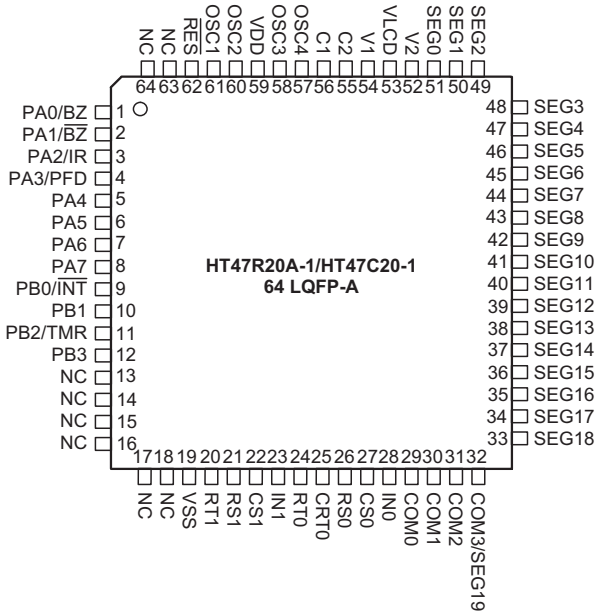
The HT47R20A-1/HT47C20-1 are 8-bit, high performance, RISC architecture microcontroller devices specifically designed for applications that interface directly to analog signals, such as those from sensors. The mask version HT47C20-1 is fully pin and functionally compatible with the OTP version HT47R20A-1 device.

The advantages of low power consumption, I/O flexibility, programmable frequency divider, timer functions, oscillator options, 2-channel RC type A/D Converter, LCD driver, HALT and wake-up functions, enhance the versatility of these devices to suit a wide range of Resistor to Frequency application possibilities such as sensor signal processing, remote metering, industrial control, consumer products, subsystem controllers, etc.

Block Diagram



Pin Assignment



Pad Description

Pad Name	I/O	Option	Function
PA0/BZ PA1/BZ PA2/IR PA3/PFD PA4~PA7	I/O	Wake-up Pull-high or None CMOS or NMOS	Bidirectional 8-bit input/output port. The low nibble of the PA can be configured as CMOS output or NMOS output with or without pull-high resistors (determined by pull-high option). NMOS output can be configured as Schmitt trigger input with or without pull-high resistors. Each bit of NMOS output can be configured as wake up input by options. Of the eight bits, PA0~PA1 can be set as I/O pins or buzzer outputs by options. PA2 can be set as an I/O pin or an IR carrier output also by options. PA3 can be set as an I/O pin or a PFD output also by options.
PB0/INT PB1 PB2/TMR PB3	I	—	4-bit Schmitt trigger input port. The PB is configured with pull-high resistors. Of the four bits, PB0 can be set as an input pin or an external interrupt input pin (INT) by software application. While PB2 can be set as an input pin or a timer/event counter input pin also by software application.
IN0 CS0 RS0 CRT0 RT0	I O O O O	—	Oscillation input pin of channel 0 Reference capacitor connection pin of channel 0 Reference resistor connection pin of channel 0 Resistor/capacitor sensor connection pin for measurement of channel 0 Resistor sensor connection pin for measurement of channel 0
IN1 CS1 RS1 RT1	I O O O	—	Oscillation input pin of channel 1 Reference capacitor connection pin of channel 1 Reference resistor connection pin of channel 1 Resistor sensor connection pin for measurement of channel 1
COM0~COM2 COM3/SEG19	O	1/2, 1/3 or 1/4 Duty	SEG19/COM3 can be set as segment or common output driver for LCD panel by options. COM0~COM2 are outputs for LCD panel plate.
SEG0~SEG18	O	—	LCD driver outputs for LCD panel segments
V1, V2, C1, C2	—	—	Voltage pump
VLCD	I	—	LCD power supply
OSC2 OSC1	O I	Crystal or RC	OSC1 and OSC2 are connected to an RC network or a crystal (by options) for the internal system clock.

Pad Name	I/O	Option	Function
OSC4 OSC3	O I	RTC or System Clock	Real time clock oscillators OSC3 and OSC4 are connected to a 32768Hz crystal oscillator for timing purposes or to a system clock source (depending on the options).
$\overline{RES}$	I	—	Schmitt trigger reset input. Active low.
VSS	—	—	Negative power supply, ground
VDD	—	—	Positive power supply
TEST1~3	I	—	Test mode input pin it disconnects in normal operation.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}–0.3V to V_{SS}+6.0V
Input Voltage ..... V_{SS}–0.3V to V_{DD}+0.3V
Storage Temperature ..... –50°C to 125°C
Operating Temperature ..... –40°C to 85°C

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
			f_{SYS}=8MHz	3.3	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC)	3V	No load, f_{SYS}=4MHz	—	0.7	1.5	mA
		5V		—	1.7	3	mA
I_{DD2}	Operating Current (RC OSC)	3V	No load, f_{SYS}=4MHz	—	0.7	1.5	mA
		5V		—	1.7	3	mA
I_{DD3}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=8MHz	—	4	8	mA
I_{DD4}	Operating Current (f_{SYS}=32768Hz)	3V	No load	—	0.25	0.5	mA
		5V		—	0.8	1.5	mA
I_{STB1}	Standby Current (*f_S=T1)	3V	No load, system HALT, LCD off at HALT	—	—	1	μA
		5V		—	—	2	μA
I_{STB2}	Standby Current (*f_S=32.768kHz OSC)	3V	No load, system HALT, LCD On at HALT, C type	—	2.5	5	μA
		5V		—	10	20	μA
I_{STB3}	Standby Current (*f_S=WDT RC OSC)	3V	No load, system HALT LCD On at HALT, C type	—	2	5	μA
		5V		—	6	10	μA
I_{STB4}	Standby Current (*f_S=32.768kHz OSC)	3V	No load, system HALT, LCD on at HALT, R type, 1/2 bias	—	17	30	μA
		5V		—	34	60	μA
I_{STB5}	Standby Current (*f_S=32.768kHz OSC)	3V	No load, system HALT, LCD on at HALT, R type, 1/3 bias	—	13	25	μA
		5V		—	28	50	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB6}	Standby Current (*f_S=WDT RC OSC)	3V	No load, system HALT, LCD on at HALT, R type, 1/2 bias	—	14	25	μA
		5V		—	26	50	μA
I_{STB7}	Standby Current (*f_S=WDT RC OSC)	3V	No load, system HALT, LCD on at HALT, R type, 1/3 bias	—	10	20	μA
		5V		—	19	40	μA
V_{IL1}	Input Low Voltage for I/O Ports, TMR and $\overline{INT}$	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR and $\overline{INT}$	3V	—	$0.7V_{DD}$	—	V_{DD}	V
		5V		$0.7V_{DD}$	—	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	3V	V_{OL}=0.1V_{DD}	6	12	—	mA
		5V		10	25	—	mA
I_{OH1}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	–2	-4	—	mA
		5V		–5	–8	—	mA
I_{OL2}	LCD Common and Segment Current	3V	V_{OL}=0.1V_{DD}	210	420	—	μA
		5V		350	700	—	μA
I_{OH2}	LCD Common and Segment Current	3V	V_{OH}=0.9V_{DD}	–80	–160	—	μA
		5V		–180	–360	—	μA
I_{OL3}	RC Oscillation Output Sink Current	3V	V_{OL}=0.3V	5	10	—	mA
I_{OH3}	RC Oscillation Output Source Current	3V	V_{OH}=2.7V	–5	–10	—	mA
R_{PH}	Pull-high Resistance of I/O Ports and $\overline{INT0}$, $\overline{INT1}$	3V	—	20	60	100	kΩ
		5V		10	30	50	kΩ
V_{LVR}	Low Voltage Reset	—	—	2.5	3.2	3.6	V
V_{LVD}	Low Voltage Detector Voltage	—		3.0	3.3	3.6	V

Note: "*" t_{SYS}= 1/f_{SYS}

"**" for power on protection

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS1}	System Clock (Crystal OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f_{SYS2}	System Clock (RC OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f_{SYS3}	System Clock (32768Hz Crystal OSC)	—	—	—	32768	—	Hz
f_{RTCOSC}	RTC Frequency	—	—	—	32768	—	Hz
f_{TIMER}	Timer I/P Frequency	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	35	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs

Functional Description

Execution Flow

The system clock for the HT47R20A-1/HT47C20-1 is derived from either a crystal or an RC oscillator. The system clock is internally divided into four non-overlapping clocks. One instruction cycle consists of four system clock cycles.

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

Program Counter – PC

The 11-bit program counter (PC) controls the sequence in which the instructions stored in the program memory are executed and its contents specify a maximum of 2048 addresses.

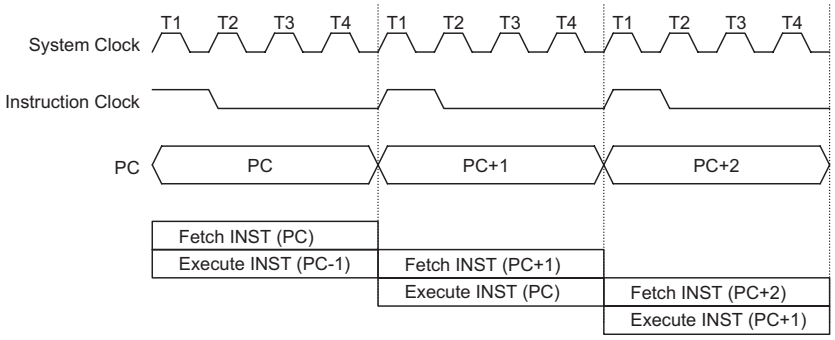
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by 1. The program counter then points to the memory word containing the next instruction code.

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

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

The lower byte of the program counter (PCL) 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.



Execution Flow

Mode	Program Counter										
	***10**	***9**	***8**	***7**	***6**	***5**	***4**	***3**	***2**	***1**	***0**
Initial reset	0	0	0	0	0	0	0	0	0	0	0
External interrupt	0	0	0	0	0	0	0	0	1	0	0
Time base interrupt	0	0	0	0	0	0	0	1	0	0	0
RTC interrupt	0	0	0	0	0	0	0	1	1	0	0
Timer/event counter interrupt	0	0	0	0	0	0	1	0	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

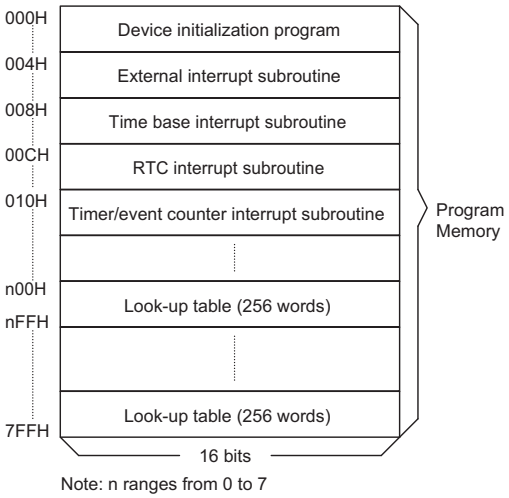
Note: *10~*0: Program counter bits
S10~S0: Stack register bits
#10~#0: Instruction code bits
@7~@0: PCL bits

Program Memory – EPROM

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

Certain locations in the program memory are reserved for special usage

- Location 000H
  This area is reserved for the initialization program. After chip reset, the program always begins execution at location 000H.
- Location 004H
  This area is reserved for the external interrupt service program. If the $\overline{INT}$ interrupt is enabled and the stack is not full, the program begins execution at location 004H.
- Location 008H
  This area is reserved for the time base interrupt service program. If time base interrupt results from a time base overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H.
- Location 00CH
  This area is reserved for the real time clock interrupt service program. If a real time clock interrupt results from a real time clock overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.
- Location 010H
  This area is reserved for the timer/event counter interrupt service program. If a 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 begins execution at location 010H.
- Table location
  Any location in the ROM space can be used as look up tables. The instructions TABRDC [m] (the current page, one 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 register (TBLH) is read only. 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 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.



Program Memory

Stack Register – 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 acknowledgment, 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

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: Table location bits
P10~P8: Current program counter bits
@7~@0: Table pointer bits

from the stack. After a chip reset, the SP will point to the top of the stack.

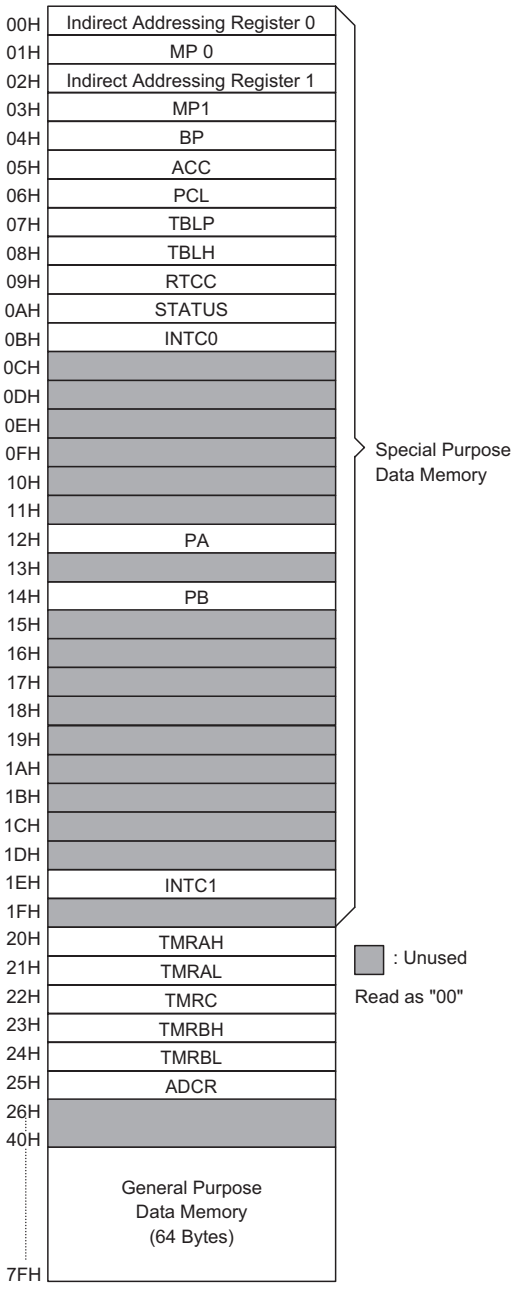
If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but the acknowledgment will be inhibited. When the stack pointer is decremented (by RET or RETI), the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use 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 are stored).

Data Memory – RAM

The data memory is designed with 85×8 bits. The data memory and is divided into two functional groups: special function registers and general purpose data memory (64×8). Most are read/write, but some are read only.

The special function registers include the indirect addressing register 0 (00H), the memory pointer register 0 (mp0; 01H), the indirect addressing register 1 (02H), the memory pointer register 1 (MP1;03H), the bank pointer (BP;04H), the accumulator (ACC;05H), the program counter lower-order byte register (PCL;06H), the table pointer (TBLP;07H), the table higher-order byte register (TBLH;08H), the real time clock control register (RTCC;09H), the status register (STATUS;0AH), the interrupt control register 0 (INTC0;0BH), the I/O registers (PA;12H, PB;14H), the interrupt control register 1 (INTC1;1EH), the timer/event counter A higher order byte register (TMRAH;20H), the timer/event counter A lower order byte register (TMRAL;21H), the timer/event counter control register (TMRC;22H), the timer/event counter B higher order byte register (TMRBH;23H), the timer/event counter B lower order byte register (TMRBL;24H), and the RC oscillator type A/D converter control register (ADCR; 25H). The remaining space before the 40H are reserved for future expanded usage and reading these location will return the result 00H. The general purpose data memory, addressed from 40H to 7FH, 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).



RAM Mapping (Bank 0)

Indirect Addressing Register

Location 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation of [00H] and [02H] access data memory pointed to by MP0 (01H) and MP1 (03H) respectively. Reading location 00H or 02H indirectly will return the result 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.

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

Accumulator

The accumulator is closely related to ALU operations. It is also mapped to location 05H of the data memory and is capable of carrying out immediate data operations. The data movement between two data memory locations must pass through the accumulator.

Arithmetic and Logic Unit – ALU

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

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

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

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 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.

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

Interrupts

The HT47R20A-1/HT47C20-1 provides an external interrupt, an internal timer/event counter interrupt, an internal time base interrupt, and an internal real time clock interrupt. The interrupt control register 0 (INTC0;0BH) and interrupt control register 1 (INTC1;1EH) both contain the interrupt control bits to set the enable or disable and interrupt request flags.

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

STATUS (0AH) Register

Bit No.	Label	Function
0	EMI	Control the master (global) interrupt (1= enabled; 0= disabled)
1	EEI	Control the external interrupt (1= enabled; 0= disabled)
2	ETBI	Control the time base interrupt (1= enabled; 0= disabled)
3	ERTI	Control the real time clock interrupt (1= enabled; 0= disabled)
4	EIF	External interrupt request flag (1= active; 0= inactive)
5	TBF	Time base request flag (1= active; 0= inactive)
6	RTF	Real time clock request flag (1= active; 0= inactive)
7	—	Unused bit, read as ″0″

INTC0 (0BH) Register

Bit No.	Label	Function
0	ETI	Control the timer/event counter interrupt (1= enabled; 0=disabled)
1~3	—	Unused bit, read as ″0″
4	TF	Internal timer/event counter request flag (1= active; 0= inactive)
5~7	—	Unused bit, read as ″0″

INTC1 (1EH) Register

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 happen during this interval, but only the interrupt request flag is recorded. If a certain interrupt needs servicing within the service routine, the EMI bit and the corresponding bit of INTC0 or INTC1 may be set allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the SP is decremented. If immediate service is desired, the stack must be prevented from becoming full.

All these kinds of interrupt have a wake-up capability. As an interrupt is serviced, a control transfer occurs by pushing the program counter onto the stack and then by branching to subroutines at specified location(s) 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.

External interrupt is triggered by a high to low transition of $\overline{\text{INT}}$ and the related interrupt request flag (EIF; bit 4 of INTC0) will be set. When the interrupt is enabled, and the stack is not full and the external interrupt is active, a subroutine call to location 04H will occur. The interrupt request flag (EIF) and EMI bits will be cleared to disable other interrupts.

The internal timer/event counter interrupt is initialized by setting the timer/event counter interrupt request flag (TF; bit 4 of INTC1), 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 10H 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 INTC0), 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.

The real time clock interrupt is initialized by setting the real time clock interrupt request flag (RTF; bit 6 of INTC0), caused by a regular real time clock signal. When the interrupt is enabled, and the stack is not full and the RTF bit is set, a subroutine call to location 0CH will occur. The related interrupt request flag (RTF) 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
External interrupt	1	04H
Time base interrupt	2	08H
Real time clock interrupt	3	0CH
Timer/event counter interrupt	4	10H

The external interrupt request flag (EIF), real time clock interrupt request flag (RTF), time base interrupt request flag (TBF), enable external interrupt bit (EEI), enable real time clock interrupt bit (ERTI), enable time base interrupt bit (ETBI), and enable master interrupt bit (EMI) constitute an interrupt control register 0 (INTC0) which is located at 0BH in the data memory. The timer/event counter interrupt request flag (TF), enable timer/event counter interrupt bit (ETI) on the other hand, constitute an interrupt control register 1 (INTC1) which is located at 1EH in the data memory. EMI, EEI, ETI, ETBI, and ERTI are used to control the enabling/disabling of interrupts. These bits prevent the requested interrupt being serviced. Once the interrupt request flags (RTF, TBF, TF, EIF) are set, they remain in the INTC1 or INTC0 respectively until the interrupts are serviced or cleared by a software instruction.

It is recommended that a program does not use the "CALL subroutine" within the interrupt subroutine. Because interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications, if only one stack is left, and enabling the interrupt is not well controlled, once the "CALL subroutine" operates in the interrupt subroutine will damage the original control sequence.

Oscillator Configuration

The HT47R20A-1/HT47C20-1 provides three oscillator circuits for system clocks, i.e., RC oscillator, crystal oscillator and 32768Hz crystal oscillator, determined by options. No matter what type of oscillator is selected, the signal is used for the system clock. The HALT mode stops the system oscillator (RC and crystal oscillator only) and ignores external signal to conserve power. The 32768Hz crystal oscillator (system oscillator) still runs at HALT mode. If the 32768Hz crystal oscillator is selected as the system oscillator, the system oscillator is not stopped; but the instruction execution is stopped. Since the (used as system oscillator or RTC oscillator) is also designed for timing purposes, the internal timing (RTC, time base, WDT) operation still runs even if the system enters the HALT mode.

Of the three oscillators, if the RC oscillator is used, an external resistor between OSC1 and is required, and the range of the resistance should be from 24kΩ to 1MΩ. The system clock, divided by 4, is available on OSC2 with pull-high resistor, which can be used to synchronize external logic. The RC oscillator provides the most cost effective solution. However, the frequency of the oscillation may vary with VDD, temperature, and the chip itself due to process variations. It is therefore, not suitable for timing sensitive operations where accurate oscillator frequency is desired.

On the other hand, if the crystal oscillator is selected, a crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for the oscillator, and no other external components are required. A resonator may be connected between OSC1 and OSC2 to replace the crystal and to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required.

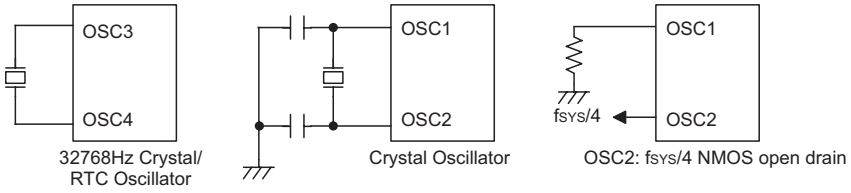
There is another oscillator circuit designed for the real time clock. In this case, only the 32.768kHz crystal oscillator can be applied. The crystal should be connected between OSC3 and OSC4.

The RTC oscillator circuit can be controlled to oscillate quickly by setting the QOSC bit (bit 4 of RTCC). It is recommended to turn on the quick oscillating function upon power on, and then turn it off after 2 seconds.

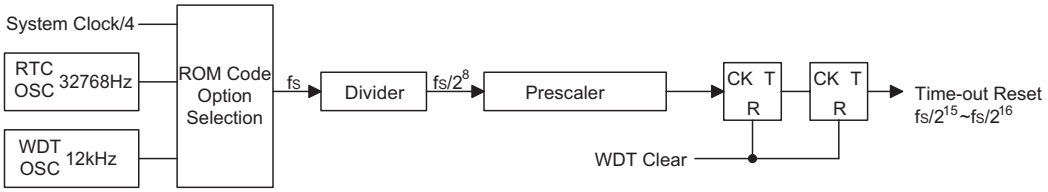
The oscillator is a free running on-chip RC oscillator, and no external components are required. Although the system enters the power down mode, the system clock stops, and the WDT oscillator still works with a period of approximately 90μs@3V. The WDT oscillator can be disabled by options to conserve power.

Watchdog Timer – WDT

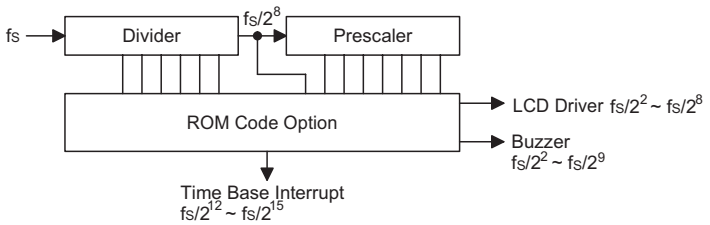
The clock source of the WDT (f_S) is implemented by a dedicated RC oscillator (WDT oscillator) or a instruction clock (system clock divided by 4) or a real time clock oscillator (RTC oscillator), decided by options. The timer is designed to prevent a software malfunction or sequence jumping to an unknown location with unpredictable results. The Watchdog Timer can be disabled by options. If the Watchdog Timer is disabled, all the executions related to the WDT result in no operation.



System Oscillator



Watchdog Timer



Time Base

If the clock source of WDT chooses the internal WDT oscillator, the time-out period may vary with temperature, VDD, and process variations. On the other hand, if the clock source selects the instruction clock and the "HALT" instruction is executed, WDT may stop counting and lose its protecting purpose, and the logic can only be restarted by external logic.

When the device operates in a noisy environment, using the on-chip RC oscillator (WDT OSC) is strongly recommended, since the HALT can cease the system clock.

The WDT overflow under normal operation will initialize "chip reset" and set the status bit TO. Whereas in the HALT mode, the overflow will initialize a "warm reset" only the program counter and SP are reset to 0. To clear the contents of WDT, three methods are adopted, external reset (a low level to $\overline{RES}$), software instruction, or a HALT instruction. The software instructions are of two types which include CLR WDT and the other set – CLR WDT1 and CLR WDT2. Of these two types of instruction, only one can be active depending on the ROM code option – "CLR WDT times selection option". If the "CLR WDT" is selected (i.e., CLR WDT times equal one), any execution of the CLR WDT instruction will clear the WDT. In case "CLR WDT1" and "CLR WDT2" are chosen (i.e. CLR WDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip because of time-out.

The WDT time-out period ranges from $2^{15}/f_S$~$2^{16}/f_S$. Because the "CLR WDT" or "CLR WDT1" and "CLR WDT2" instruction only clear the last two-stage of the WDT.

Multi-function Timer

The HT47R20A-1/HT47C20-1 provides a multi-function timer for WDT, time base and real time clock but with different time-out periods. The multi-function timer consists of a 8-stage divider and an 7-bit prescaler, with the clock source coming from WDT OSC or RTC OSC or the instruction clock (i.e., system clock divided by 4). The multi-function timer also provides a selectable frequency signal (ranges from $f_S/2^2$ to $f_S/2^8$) for LCD driver circuits, and a selectable frequency signal (ranges from $f_S/2^2$ to $f_S/2^9$) for buzzer output by options. It is recommended to select a 4kHz signal for LCD driver circuits for proper display.

Time Base

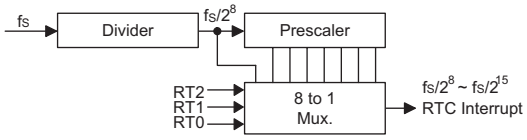
The time base offers a periodic time-out period to generate a regular internal interrupt. Its time-out period ranges from $f_S/2^{12}$ to $f_S/2^{15}$ selected by options. If time base time-out occurs, the related interrupt request flag (TBF; bit 5 of INTC0) is set. But if the interrupt is enabled, and the stack is not full, a subroutine call to location 08H occurs.

When the HALT instruction is executed, the time base still works (if WDT clock source comes from WDT RC OSC or RTC OSC) and can wake up from HALT mode.

If the TBF is set "1" before entering the HALT mode, the wake up function will be disabled.

Real Time Clock – RTC

The real time clock (RTC) is operated in the same manner as the time base that is used to supply a regular internal interrupt. Its time-out period ranges from $f_S/2^8$ to $f_S/2^{15}$ by software programming. Writing data to RT2, RT1 and RT0 (bits 2, 1, 0 of RTCC;09H) yields various



Real Time Clock

time-out periods. If the RTC time-out occurs, the related interrupt request flag (RTF; bit 6 of INTC0) is set. But if the interrupt is enabled, and the stack is not full, a subroutine call to location 0CH occurs. The real time clock time-out signal can also be applied to be a clock source of timer/event counter, so as to get a longer time-out period.

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

Note: ″*″ not recommended to be used

Power Down Operation – HALT

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

- The system oscillator will turn off but the WDT oscillator or RTC oscillator keeps running (if the WDT oscillator or the real time clock is selected).
- The contents of the on-chip RAM and registers remain unchanged.
- The WDT will be cleared and recounted again (if the WDT clock comes from the WDT oscillator or the real time clock oscillator).
- All I/O ports maintain their original status.
- The PDF flag is set and the TO flag is cleared.
- LCD driver is still running by ROM code option (if the WDT OSC or RTC OSC is selected).

The system can leave the HALT mode by means of an external reset, an interrupt, an external falling edge signal on port A or a overflow. An external reset causes a device initialization and the WDT overflow performs a ″warm reset″. Examining the TO and PDF flags, the reason for chip reset can be determined. The PDF flag is cleared when the system power-up or executing the CLR WDT instruction and is set when the HALT instruction is executed. The TO flag is set if a WDT time-out occurs, it causes a wake-up that only resets the program counter and SP, the others maintain their original status.

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake up the device by ROM code option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If awakening from an interrupt, two sequences may happen. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program will resume execution at the next instruction. If the interrupt is enabled and the stack is not full, the regular interrupt response takes place.

If an interrupt request flag is set to ″1″ before entering the HALT mode the wake-up function of the related interrupt will be disabled.

Once a wake-up event occurs, it takes 1024 (system clock period) to resume normal operation. In other words, a dummy period will be inserted after the wake-up. If the wake-up results from an interrupt acknowledgment, the actual interrupt subroutine execution is delayed by one more cycle. If the wake-up results in the next instruction execution, this will execute immediately after a dummy period has finished.

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

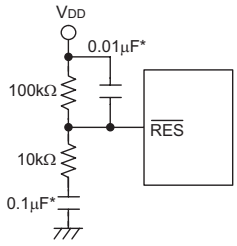
Reset

- There are three ways in which a reset may occur.
- $\overline{\text{RES}}$ reset during normal operation
- $\overline{\text{RES}}$ reset during HALT
- WDT time-out reset during normal operation

The WDT time-out during HALT is different from other chip reset conditions, since it can perform a warm reset that just resets the program counter and SP leaving the other circuits in their original state. Some registers remain unchanged during any 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	$\overline{\text{RES}}$ reset during power-up
u	u	$\overline{\text{RES}}$ reset during normal operation
0	1	$\overline{\text{RES}}$ wake-up HALT
1	u	WDT time-out during normal operation
1	1	WDT wake-up HALT

Note: ″u″ means ″unchanged″.



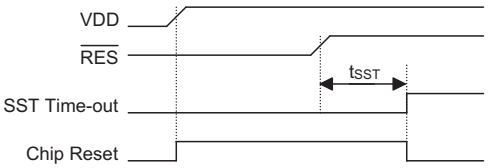
Reset Circuit

Note: ″*″ Make the length of the wiring, which is connected to the $\overline{\text{RES}}$ pin as short as possible, to avoid noise interference.

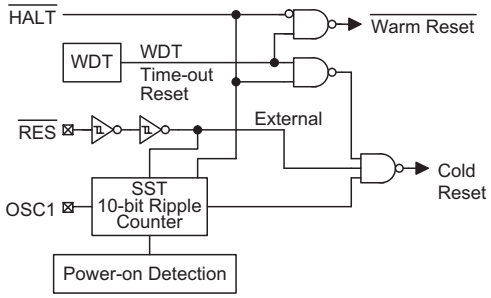
To guarantee that the system oscillator has started and stabilized, the SST (System Start-up Timer) provides an extra delay. There is an extra delay of 1024 system clock pulses when the system awakes from the HALT state or when the system powers up.

The functional unit chip reset status are shown below.

Program Counter	000H
Interrupt	Disabled
Prescaler, Divider	Cleared
WDT, RTC, Time Base	Clear. After master reset, begin counting
Timer/Event Counter	Off
Input/output ports	Input mode
SP	Points to the top of the stack



Reset Timing Chart



Reset Configuration

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

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)
TMRAH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRAL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	0000 1---	0000 1---	0000 1---	0000 1---	uuuu u---
TMRBH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRBL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	1xxx --00	1xxx --00	1xxx --00	1xxx --00	uuuu --uu
Program Counter	000H	000H	000H	000H	000H*
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	---0 ---0	---0 ---0	---0 ---0	---0 ---0	---u ---u
RTCC	--00 0111	--00 0111	--00 0111	--00 0111	--uu uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu

Note: "*" refers to warm reset
"u" means unchanged
"x" means unknown

Timer/Event Counter

One 16-bit timer/event counter with PFD output or two channels of RC type A/D converter is implemented in the HT47R20A-1/HT47C20-1. The ADC/TM bit (bit 1 of ADCR register) decides whether timer A and timer B is composed of one 16-bit timer/event counter or timer A and timer B composed of two channels RC type A/D converter.

The TMRAL, TMRAH, TMRBL, TMRBH compose 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.

Using the internal clock, there are three reference time base. The timer/event counter internal clock source may come from the system clock or system clock/4 or RTC time-out signal to generator an accurate time base.

Using 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 TMRBL only writes the data into a low byte buffer, and writing 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 TMRBH operations and writing TMRBL will keep the timer/event counter preload register unchanged.

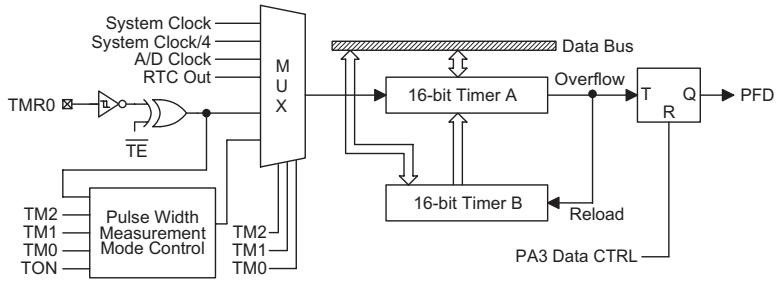
Reading TMRAH will also latch the TMRAL into the low byte buffer to avoid the false timing problem. Reading TMRAL returns the contents of the low byte buffer. In other words, the low byte of the timer/event counter can not be read directly. It must read the TMRAH first to make the low byte contents of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

If the timer/event counter is on, the TMRAH, TMRAL, TMRBH and TMRBL cannot be read or written. To avoid conflicting between timer A and timer B, the TMRAH, TMRAL, TMRBH and TMRBL registers should be accessed with ″MOV″ instruction under timer off condition.

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 makes the starting value be placed in the timer/event counter preload register, while reading timer A yields the contents of the timer/event counter. Timer B is timer/event counter preload register.



Timer/Event Counter

Bit No.	Label	Function
0~2	—	Unused bit, read as ″0″
3	TE	Defines the TMR active edge of timer/event counter (0=active on low to high; 1=active on high to low)
4	TON	Enable or disable timer counting (0=disable; 1=enable)
5 6 7	TM0 TM1 TM2	Defines the operating mode (TM2, TM1, TM0) 000=Timer mode (system clock) 001=Timer mode (system clock/4) 010=Timer mode (RTC output) 011=A/D clock mode (RC oscillation decided by ADCR register) 100=Event counter mode (external clock) 101=Pulse width measurement mode (system clock/4) 110=Unused 111=Unused

TMRC (22H) Register

The TM0, TM1 and TM2 bits define the operation mode. The event count mode is used to count external events, which means that the clock source comes from an external (TMR) pin. The A/D clock mode is used to count external A/D clock, the RC oscillation mode is decided by ADCR register. The timer mode functions as a normal timer with the clock source coming from the internal selected clock source. Finally, the pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR). The counting is based on the instruction clock.

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 generates the corresponding interrupt request flag (TF; bit 4 of INTC1) at the same time.

In the pulse width measurement mode with the TON and TE bits are equal to 1, once the TMR has received a transient from low to high (or high to low if the TE bit is 0) it will start counting until the TMR returns to the original level and resets the TON. The measured result 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 overflows, the counter is reloaded from the timer/event counter preload register and issues interrupt request just like the other three 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 be automatically cleared after the measurement cycle is completed. But in the other three modes, the TON can only be reset by instructions. The overflow of the timer/event counter is one of the wake-up sources and can also be applied to a PFD (Programmable Frequency Divider) output at PA3 by options. No matter what the operation mode is, writing a 0 to ETI can disable the corresponding interrupt service. When the PFD function is selected, executing ″CLR PA.3″ instruction to enable PFD output and executing ″SET PA.3″ instruction to disable PFD output and PA.3 output low level.

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 result in a counting error, this must be taken into consideration by the programmer.

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

If the timer/event counter is on, the TMRAH, TMRAL, TMRBH and TMRBL cannot be read or written. Only when the timer/event counter is off and when the instruction ″MOV″ is used could those four registers be read or written.

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

```
clr tmrc
clr adcr.1                  ; set timer mode
clr intc1.4                 ; 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, 00110000b            ; timer clock source=T1 and timer on
mov tmrc, a

P10:
clr wdt
snz intcl.4                 ; polling timer/event counter interrupt request flag

P10
clr intcl.4                 ; clear timer/event counter interrupt request flag
                            ; program continue
```

A/D Converter

Two channels of RC type A/D converter are implemented in the HT47R20A-1/HT47C20-1. The A/D converter contains two 16-bit programmable count-up counter and the Timer A clock source may come from the system clock, instruction clock or RTC output. The timer B clock source may come from the external RC oscillator. The TMRAL, TMRAH, TMRBL, TMRBH is composed of the A/D converter when ADC/$\overline{TM}$ bit (bit 1 of ADRC register) is ″1″.

The A/D converter timer B clock source may come from channel 0 (IN0 external clock input mode, RS0~CS0 oscillation, RT0~CS0 oscillation, CRT0~CS0 oscillation (CRT0 is a resistor), or RS0~CRT0 oscillation (CRT0 is a capacitor) or channel 1 (RS1~CS1 oscillation, RT1~CS1 oscillation or IN1 external clock input). The timer A clock source is from the system clock, instruction clock or RTC prescaler clock output decided by TMRC register.

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 OVB/$\overline{OVA}$ bit (bit 0 of ADCR register) decides whether timer A overflows 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/timer B and reading TMRAH/TMRBH gets the contents of the timer A/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/timer B (16-bit) simultaneously. The timer A/timer B is changed by writing TMRAH/TMRBH operations and writing TMRAL/TMRBL will keep timer A/timer B unchanged.

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

If the A/D converter timer A and timer B are counting, the TMRAH, TMRAL, TMRBH and TMRBL cannot be read or written. To avoid conflicting between timer A and timer B, the TMRAH, TMRAL, TMRBH and TMRBL registers should be accessed with ″MOV″ instruction under timer A and timer B off condition.

The bit4~bit7 of ADCR decides which resistor and capacitor compose an oscillation circuit and input to TMRBH and TMRBL.

The TM0, TM1 and TM2 bits of TMRC define the clock source of timer A. It is recommended that the clock source of timer A use the system clock, instruction clock or RTC prescaler clock.

The TON bit (bit 4 of TMRC) is set ″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 INTC1) 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. Only when the timer/event counter is off and when the instruction ″MOV″ is used could those four registers be read or written.

Bit No.	Label	Function
0	OVB/$\overline{OVA}$	In the RC type A/D converter mode, this bit is used to define the timer/event counter interrupt which comes from timer A overflow or timer B overflow. (0= timer A overflow; 1= timer B overflow) In the timer/event counter mode, this bit is void.
1	ADC/$\overline{TM}$	Defines 16 timer/event counters or RC type A/D converter is enabled. (0= timer/event counter enable; 1= A/D converter is enabled)
2~3	—	Unused bit, read as ″0″.
4 5 6 7	M0 M1 M2 M3	Defines the A/D converter operating mode (M3, M2, M1, M0) 0000= IN0 external clock input mode 0001= RS0~CS0 oscillation (reference resistor and reference capacitor) 0010= RT0~CS0 oscillation (resistor sensor and reference capacitor) 0011= CRT0~CS0 oscillation (resistor sensor and reference capacitor) 0100= RS0~CRT0 oscillation (reference resistor and sensor capacitor) 0101= RS1~CS1 oscillation (reference resistor and reference capacitor) 0110= RT1~CS1 oscillation (resistor sensor and reference capacitor) 0111= IN1 external clock input mode 1XXX= Unused mode

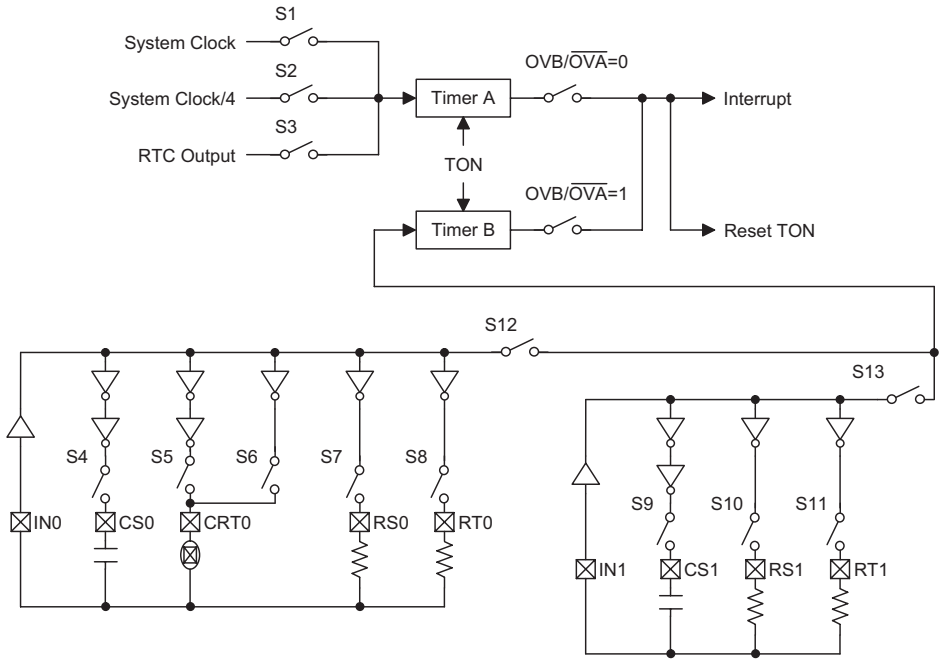
ADCR (25H) Register

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

```
clr tmrc
clr adcr.1                  ; set timer mode
clr intc1.4                 ; 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, 00010010b            ; RS0~CS0; 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, 00110000b            ; timer A clock source=T1 and timer on
mov tmrc, a

p10:
clr wdt
snz intcl.4                 ; polling timer/event counter interrupt request flag
jmp p10
clr intcl.4                 ; clear timer/event counter interrupt request flag
                            ; program continue
```

Example for RC type AD converter mode (Timer B overflow):

```
clr tmrc
clr adcr.1                  ; set timer mode
clr intc1.4                 ; 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, 00010011b            ; RS0~CS0; set RC type ADC mode; set Timer B overflow
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=T1 and timer on
mov tmrc, a

p10:
clr wdt
snz intcl.4                 ; polling timer/event counter interrupt request flag
jmp p10

clr intcl.4                 ; clear timer/event counter interrupt request flag
                            ; program continue
```



TN2	TN1	TN0	S1	S2	S3
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1
Other			0	0	0

Note: 0=off, 1=on

M3	M2	M1	M0	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	0	0	1	0	0	0	0	1	0
0	0	1	0	1	0	0	0	1	0	0	0	1	0
0	0	1	1	1	0	1	0	0	0	0	0	1	0
0	1	0	0	0	1	0	1	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1	1	0	0	1
0	1	1	0	0	0	0	0	0	1	0	1	0	1
0	1	1	1	0	0	0	0	0	0	0	0	0	1
1				0	0	0	0	0	0	0	0	0	0

Note: 0=off, 1=on

RC Type A/D Converter

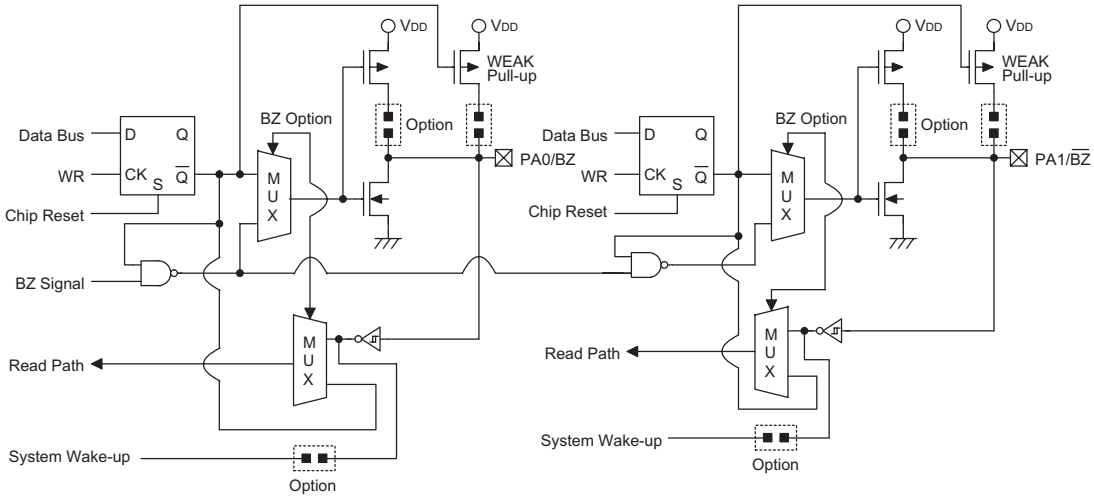
Input/Output Ports

There are 8-bit bidirectional input/output port and 4-bit input port in the HT47R20A-1/HT47C20-1, labeled PA and PB which are mapped to the data memory of [12H] and [14H] respectively. The high nibble of the PA is NMOS output and input with pull-high resisters. The low nibble of the PA can be used for input/output or output operation by selecting NMOS or CMOS output by options. Each bit on the PA can be configured as a wake-up input, and the low nibble of the PA with or without pull-high resistor by options. PB can only be used for input operation, and each bit is with pull high resistor. Both are for the input operation, these ports are non-latched, that is, the inputs should be ready at the T2 rising edge of the instruction "MOV A, [m]" (m=12H or 14H). For PA output operation, all data are latched and remain unchanged until the output latch is rewritten.

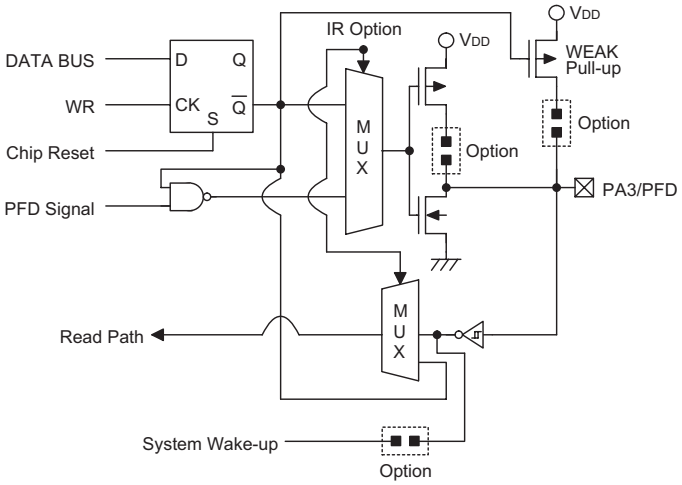
When the structures of PA are open drain NMOS type, it should be noted that, before reading data from the pads, a "1" should be written to the related bits to disable the NMOS device. That is done first before executing the instruction "MOV A, 0FFH" and "MOV [12H], A" to disable related NMOS device, and then "MOV A, [12H]" to get stable data.

After chip reset, these input lines remain at a high level or are left floating (by ROM code option).

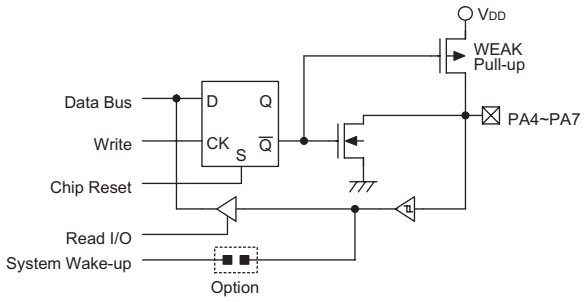
Some instructions first input data and then follow the output operations. For example, "SET [m].i", "CLR [m].i", "CPL [m]", "CPLA [m]" read the entire port states into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or to the accumulator. Each bit of the PA output latches can not use these instruction, which may change the input lines to output lines (when input line is at low level).



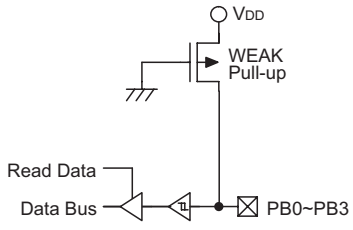
PA0/BZ, PA1/$\overline{BZ}$ Input/Output Port



PA2/IR, PA3/PFD Input/Output Port



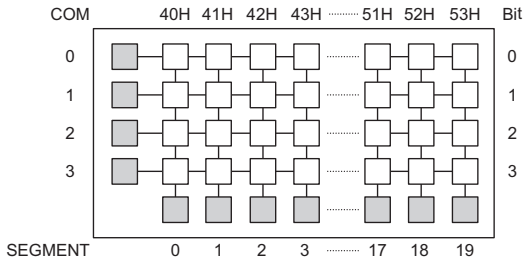
PA4~PA7 Input/Output Ports



PB Input Lines

LCD Display Memory

The HT47R20A-1/HT47C20-1 provides an area of embedded data memory for LCD display. The LCD display memory is designed into 20×3 bits. If the LCD selected 19×4 segments output, the 53H of the LCD display memory can not be accessed. This area is located from 40H to 53H of the RAM at Bank 1. Bank pointer (BP; located at 04H of the data memory) is the switch between the general data memory and the LCD display memory. When the BP is set "1" any data written into 40H~53H will effect the LCD display (indirect addressing mode using MP1). When the BP is cleared "0", any data written into 40H~53H means to access the general purpose data memory. The LCD display memory can be read and written 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 HT47R20A-1/HT47C20-1.

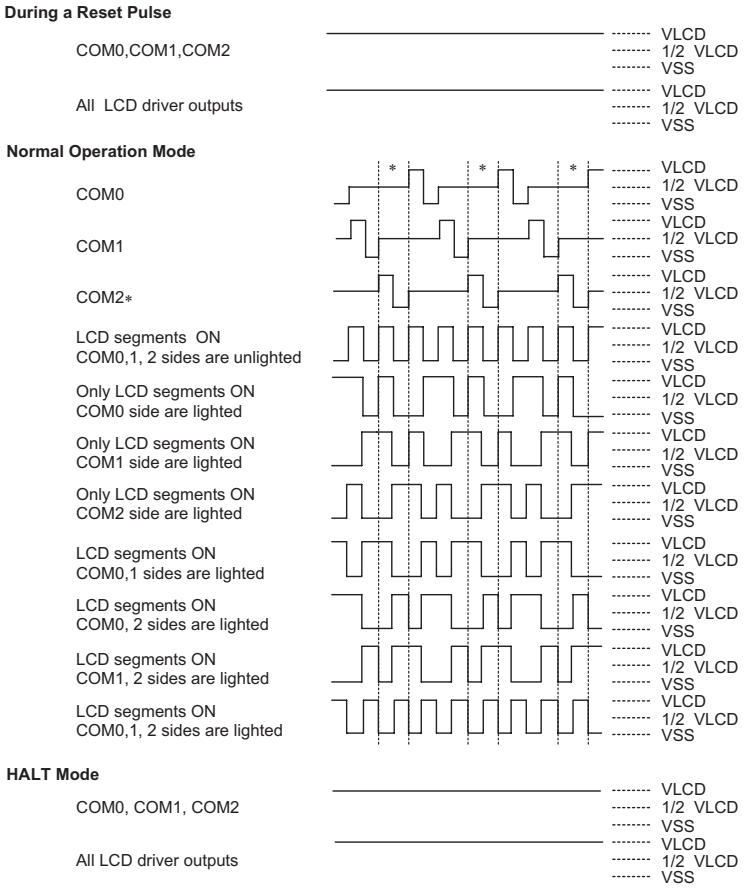


Display Memory (Bank 1)

LCD Driver Output

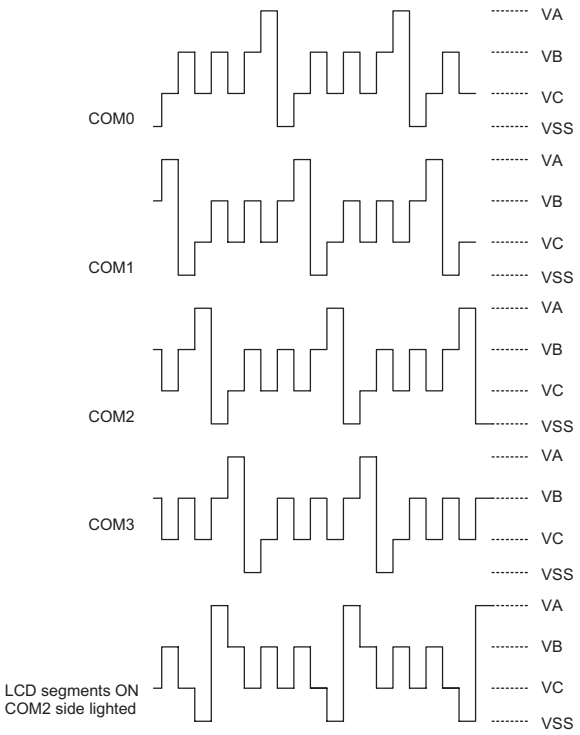
The output number of the HT47R20A-1/HT47C20-1 LCD driver can be 20×2, 20×3 or 19×4 by options (i.e., 1/2 duty, 1/3 duty or 1/4 duty).

The bias type of the LCD driver can be "C" type or "R" type. A capacitor mounted between C1 and C2 pins is needed. The bias voltage of the LCD driver can be 1/2 bias or 1/3 bias by options. If 1/2 bias is selected, a capacitor mounted between V2 pin and the ground is required. If 1/3 bias is selected, two capacitors are needed for V1 and V2 pins. Refer to the application diagram. If the "R" bias type is selected, no external capacitor is required.



Note: "*" Omit the COM2 signal, if the 1/2 duty LCD is used.

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



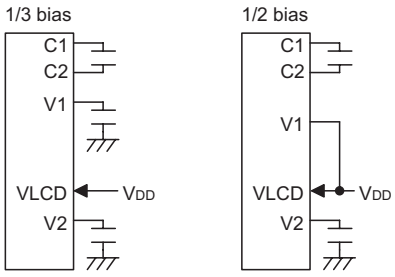
Note: 1/4 duty, 1/3 bias, C type: "VA" 3/2 VLCD, "VB" VLCD, "VC" 1/2 VLCD
1/4 duty, 1/3 bias, R type: "VA" VLCD, "VB" 2/3 VLCD, "VC" 1/3 VLCD

LCD Driver Output

Low Voltage Reset/Detector Functions

There is a low voltage detector (LVD) and a low voltage reset circuit (LVR) implemented in the microcontrollers. These two functions can be enabled/disabled by ROM code options. The LVD can be enabled/disabled by ROM code options. Once the ROM code options of LVD is enabled, the user can use the RTCC.3 to enable or disable (1/0) the LVD circuit and read the LVD detector status (0/1) from RTCC.5; otherwise, the LVD function is disabled.

The LVR has the same effect or function with the external $\overline{RES}$ signal which performs chip reset. During HALT state, LVR is disabled.



V1, V2, VLCD Application Diagram (C Type)

The definitions of RTCC register are listed in the following table.

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

Note: ″*″ Once the function is enabled the reference generator should be enabled; otherwise the reference generator is controlled by LVR ROM code option.

RTCC (09H) Register

Buzzer

HT47R20A-1/HT47C20-1 provides a pair of buzzer output BZ and $\overline{BZ}$, which share pins with PA0 and PA1 respectively, as determined by options. Its output frequency can also be selected by options.

When the buzzer function is selected, setting PA.0 and PA.1 ″0″ simultaneously will enable the buzzer output and setting PA.0 ″1″ will disable the buzzer output and setting PA.0 ″0″ and PA.1 ″1″ will only enable the BZ output and disable the $\overline{BZ}$ output.

PA1	PA0	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

Buzzer Enable

IR Carrier

HT47R20A-1/HT47C20-1 provides carrier driving capability that allows for easy interfacing to an infrared diode, which share pin with PA2, as determined by options.

When the carrier option is selected, setting PA2 ″0″ (″CLR PA.2″) will enable the carrier output and setting PA2 ″1″ (″SET PA.2″) will disable the carrier output and the PA2 output is at low level. The IR carrier frequency is system clock divided by 12 and it is 1/4 duty.

PA2	Function
0 (CLR PA.2)	PA2=IR carrier output
1 (SET PA.2)	PA2=0

Programmable Frequency Divider – PFD

The PFD output shares pin with PA3, as determined by options.

When the PFD option is selected, setting PA3 ″0″ (″CLR PA.3″) will enable the PFD output and setting PA3 ″1″ (″SET PA.3″) will disable the PFD output and PA3 output at low level.

PFD output frequency =

$$\frac{1}{2}\times\frac{1}{\text{timer overflow period}}$$

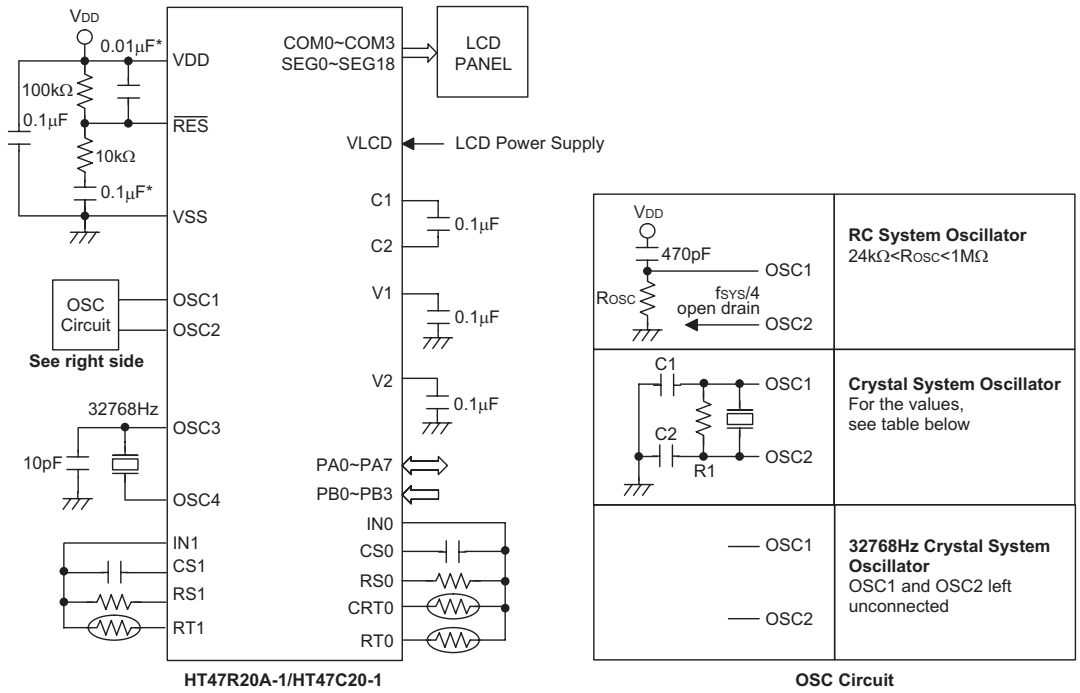
PA3	Function
0 (CLR PA.3)	PA3=PFD output
1 (SET PA.3)	PA3=0

Option

The following shows many kinds of options in the HT47R20A-1/HT47C20-1. All these options should be defined in order to ensure proper system functioning.

No.	Option
1	OSC type selection. This option is to decide if an RC, a crystal oscillator or RTC oscillator is chosen as system clock.
2	Clock source selection of WDT, RTC and Time Base. There are three types of selection: system clock/4 or RTC OSC or WDT OSC.
3	WDT enable or disable selection. WDT can be enabled or disabled.
4	CLR WDT times selection. This option defines how to clear the WDT by instruction. One time means that the "CLR WDT" can clear the WDT. "Two times" means that only if both of the "CLR WDT1" and "CLR WDT2" have been executed, then WDT can be cleared.
5	Time Base time-out period selection. The Time Base time-out period ranges from $f_S/2^{12}$ to $f_S/2^{15}$. "f_S" means the clock source of WDT.
6	Buzzer output frequency selection. There are eight types frequency signals for buzzer output: $f_S/2^2$~$f_S/_2{}^9$. "f_S" means the clock source of WDT.
7	Wake-up selection. This option defines the wake-up function activity. External I/O pins (PA only) all have the capability to wake-up the chip from a HALT mode by a following edge.
8	Pull high selection. This option is to decide whether the pull high resistance is viable or not on the low nibble of the PA.
9	PA CMOS or NMOS selection. The structure of the low nibble of the PA can be selected to be CMOS or NMOS. When the CMOS is selected, the related pins only can be used for output operations. When the NMOS is selected, the related pins can be used for input or output operations.
10	I/O pins share with other function selection. PA0/BZ, PA1/$\overline{BZ}$: PA0 and PA1 can be set as I/O pins or buzzer outputs. PA2/IR: PA2 can be set as I/O pins or IR carrier output. PA3/PFD: PA3 can be set as I/O pins or PFD output.
11	LCD common selection. There are three types of selection: 2 common (1/2 duty) or 3 common (1/3 duty) or 4 common (1/4 duty). If the 4 common is selected, the segment output pin "SEG32" will be set as a common output.
12	LCD driver clock selection. There are seven types of frequency signals for the LCD driver circuits: $f_S/2^2$~$f_S/2^8$. "f_S" means the clock source of WDT.
13	LCD on or LCD off at the HALT mode selection. The LCD can be enable or disable at the HALT mode.
14	LVD enable or disable
15	LVR enable or disable

Application Circuits



Note: The resistance and capacitance for reset circuit should be designed in such a way as to ensure that the VDD is stable and remains within a valid operating voltage range before bringing $\overline{RES}$ to high.

"*" Make the length of the wiring, which is connected to the $\overline{RES}$ pin as short as possible, to avoid noise interference.

The following table shows the C1, C2 and R1 value according different crystal values. (For reference only)

Crystal or Resonator	C1, C2	R1
4MHz Crystal	0pF	10kΩ
4MHz Resonator (3 pin)	0pF	12kΩ
4MHz Resonator (2 pin)	10pF	12kΩ
3.58MHz Crystal	0pF	10kΩ
3.58MHz Resonator (2 pin)	25pF	10kΩ
2MHz Crystal & Resonator (2 pin)	25pF	10kΩ
1MHz Crystal	35pF	27kΩ
480kHz Resonator	300pF	9.1kΩ
455kHz Resonator	300pF	10kΩ
429kHz Resonator	300pF	10kΩ
The function of the resistor R1 is to ensure that the oscillator will switch off should low voltage conditions occur. Such a low voltage, as mentioned here, is one which is less than the lowest value of the MCU operating voltage. Note however that if the LVR is enabled then R1 can be removed.		

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

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

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

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

Instruction Definition

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

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

Operation: ACC ← ACC + [m] + C

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

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

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

Operation: [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation: ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation: ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation: [m] ← ACC + [m]

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

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

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

Operation: ACC ← ACC ″AND″ [m]

Affected flag(s): Z

AND A,x Logical AND immediate data to ACC

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

Operation: ACC ← ACC ″AND″ x

Affected flag(s): Z

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

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

Operation: [m] ← ACC ″AND″ [m]

Affected flag(s): Z

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None
CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CPL [m] Complement Data Memory

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

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

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

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

HALT Enter power down mode

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

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

Affected flag(s) TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z
INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z
JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None
MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None
MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None
NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x Logical OR immediate data to ACC

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

Operation: ACC ← ACC ″OR″ x

Affected flag(s): Z

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

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

Operation: [m] ← ACC ″OR″ [m]

Affected flag(s): Z

RET Return from subroutine

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

Operation: Program Counter ← Stack

Affected flag(s): None

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

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

Operation: Program Counter ← Stack
ACC ← x

Affected flag(s): None

RETI Return from interrupt

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

Operation: Program Counter ← Stack
EMI ← 1

Affected flag(s): None

RL [m] Rotate Data Memory left

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

Operation $[m] \leftarrow [m] - 1$
Skip if $[m] = 0$

Affected flag(s) None

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

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

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

Affected flag(s) None

SET [m] Set Data Memory

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

Operation $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation $[m].i \leftarrow 1$

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m] = 0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC = 0

Affected flag(s) None

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

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

Operation Skip if [m].i $\neq$ 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.

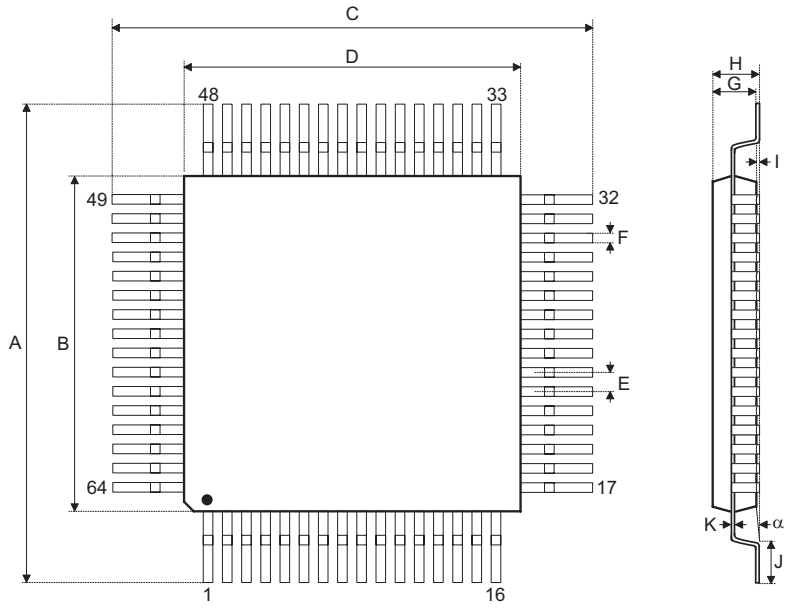
Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

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

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



Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	8.9	—	9.1
B	6.9	—	7.1
C	8.9	—	9.1
D	6.9	—	7.1
E	—	0.4	—
F	0.13	—	0.23
G	1.35	—	1.45
H	—	—	1.6
I	0.05	—	0.15
J	0.45	—	0.75
K	0.09	—	0.20
α	0°	—	7°

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

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

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

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