HOLTEK

HT46C20
8-Bit Microcontroller

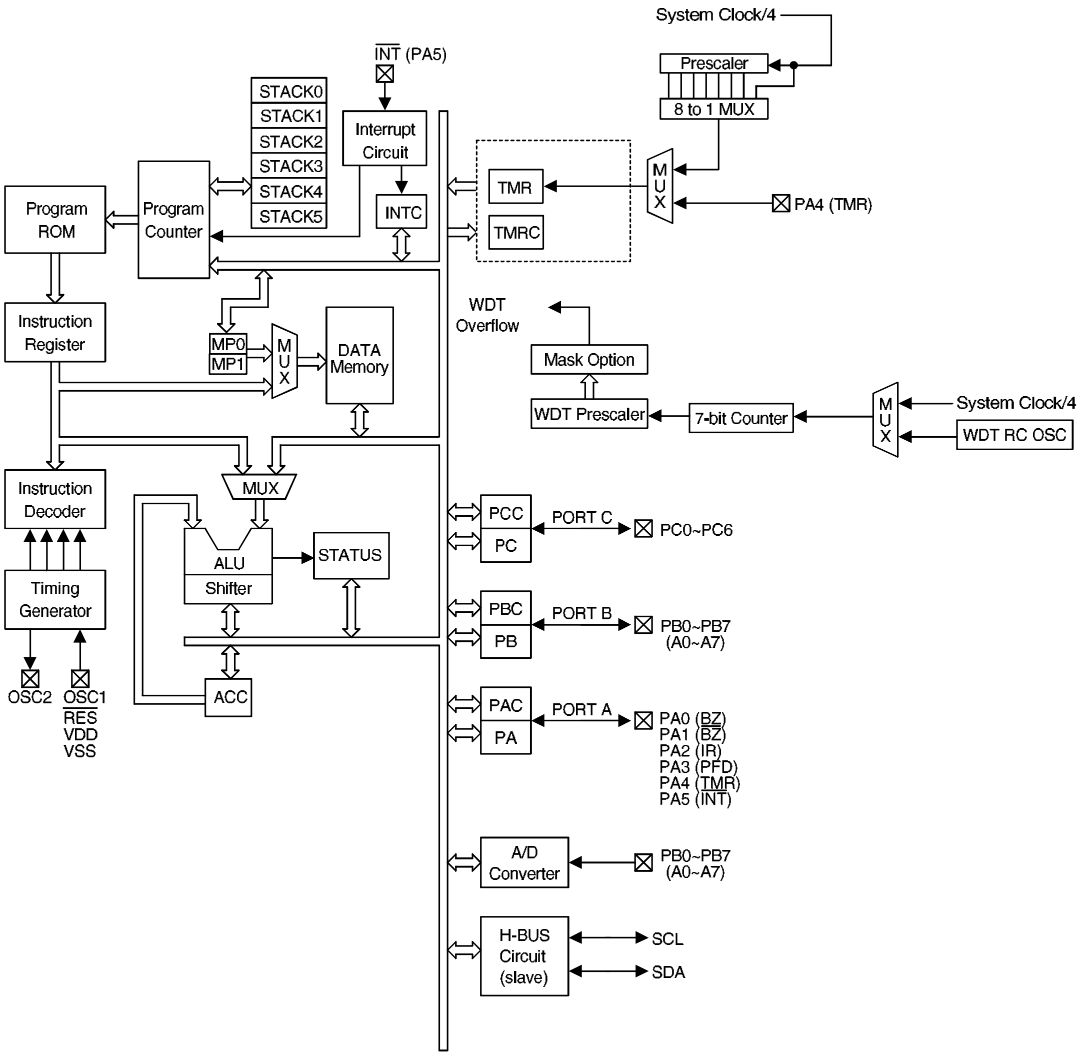
Features

- Operating voltage: 4.5V~5.5V
- 21 bidirectional I/O lines
- One interrupt input
- One 8-bit programmable timer/event counter with PFD (programmable frequency divider) and overflow interrupt
- On-chip crystal and RC oscillator
- Watchdog timer
- 2K × 14 program memory ROM
- 64 × 8 data memory RAM
- Halt function and wake-up feature reduce power consumption
- 63 powerful instructions
- Up to 667ns instruction cycle with 6MHz system clock at V_{DD}=5V
- All instructions in one or two machine cycles
- 14-bit table read instructions
- Six-level subroutine nesting
- Bit manipulation instructions
- One buzzer output
- One 38kHz or 40kHz IR carrier output (455kHz or 480kHz system clock only)
- 10-bit A/D converter
- H-BUS (slave mode)

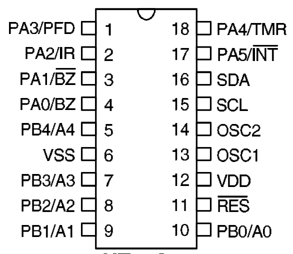
General Description

The HT46C20 is an 8-bit high performance RISC-like microcontroller specifically designed for multiple I/O with A/D converter product applications. The device is particularly suitable for use in products such as remote controllers, fan/light controllers, washing machine controllers, scales, toys and various subsystem controllers. A halt feature is included to reduce power consumption.

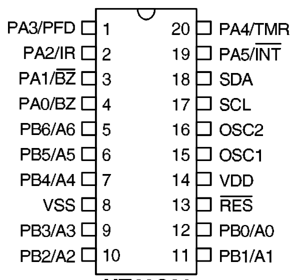
Block Diagram



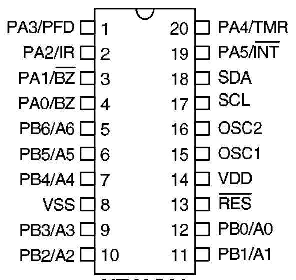
Pin Assignment



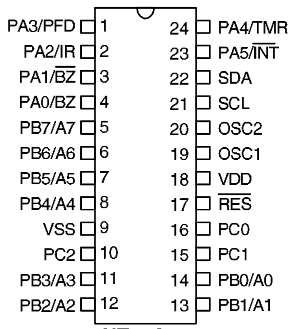
HT46C20
– 18 DIP-A-0



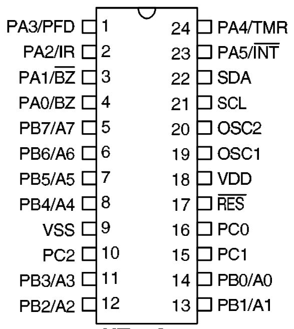
HT46C20
– 20 DIP-A-0



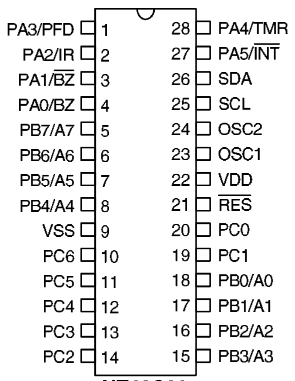
HT46C20
– 20 SOP-A-0



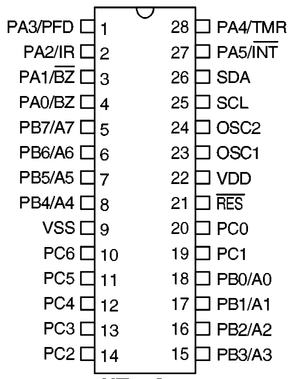
HT46C20
– 24 SKDIP-A-0



HT46C20
– 24 SOP-A-0

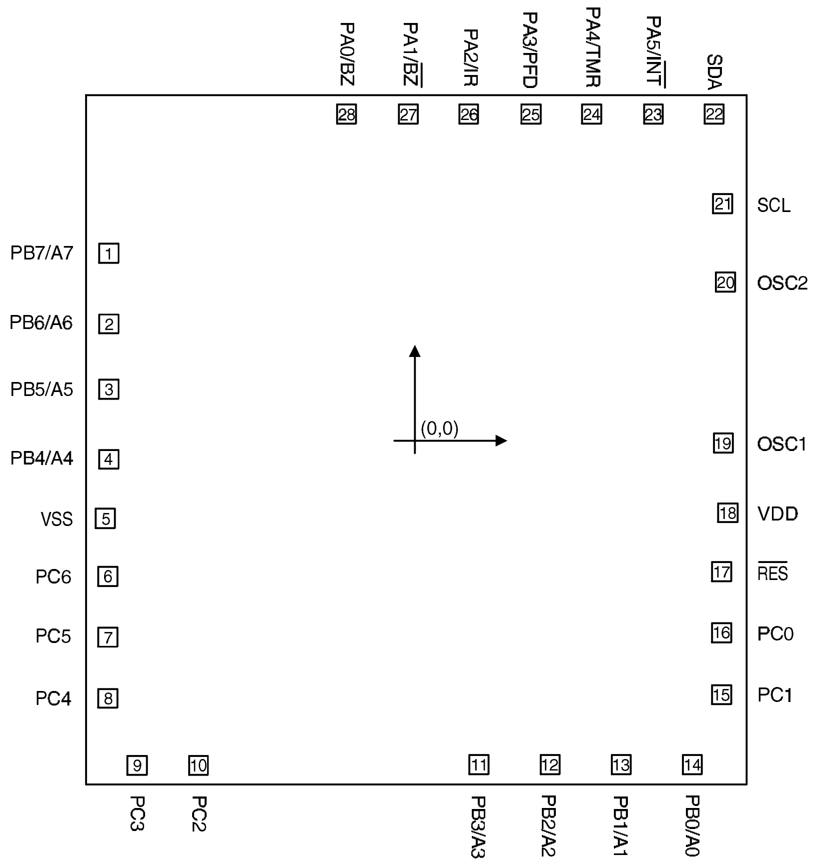


HT46C20
– 28 SKDIP-A-0



HT46C20
– 28 SOP-A-0

Pad Assignment



Chip size: 3220 × 3450 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

* The TMR pad is floating when PA4 input pin mask option is selected, non pull-high resistor.

* The $\overline{\text{INT}}$ pad is floating when PA5 input pin mask option is selected, non pull-high resistor.

Pad Coordinates

Unit: µm

Pad No.	X	Y	Pad No.	X	Y
1	–1397.50	856.25	15	1401.10	–1160.15
2	–1397.50	533.85	16	1401.10	–877.35
3	–1397.50	235.65	17	1401.10	–599.95
4	–1397.50	–86.75	18	1429.70	–329.65
5	–1414.20	–356.15	19	1409.40	–11.55
6	–1401.70	–620.45	20	1418.90	724.85
7	–1401.70	–897.05	21	1405.90	1083.85
8	–1401.70	–1176.75	22	1367.20	1491.55
9	–1268.60	–1483.95	23	1090.10	1491.55
10	–987.80	–1483.95	24	807.50	1491.55
11	293.20	–1480.35	25	530.10	1491.55
12	617.20	–1480.35	26	247.30	1491.55
13	943.80	–1480.35	27	–30.10	1491.55
14	1267.80	–1480.35	28	–312.90	1491.55

Pin Description

Pin Name	I/O	Mask Option	Function
PA0/BZ PA1/$\overline{BZ}$ PA2/IR PA3/PFD PA4/TMR PA5/$\overline{INT}$	I/O	PA0 or BZ PA1 or $\overline{BZ}$ PA2 or IR PA3 or PFD Wake-up Pull-high or None CMOS or NMOS	Bidirectional 6-bit input/output port. Each bit can be configured as wake-up input by mask option. Software instructions determine the CMOS (or NMOS by mask option) output or schmitt trigger input with or without pull-high resistors (mask option). PA0~PA1 can be set as I/O pins or buzzer output by mask option. PA2 can be set as an I/O pin or a IR carrier output also by mask option. PA3 can be set as an I/O pin or a PFD output also by mask option. PA4 can be set as an I/O pin or a timer/event counter input pin also by software application. External interrupt schmitt trigger input in PA5. Edge triggered activated on a high to low transition.
PB0/A0 PB1/A1 PB2/A2 PB3/A3 PB4/A4 PB5/A5 PB6/A6 PB7/A7	I/O	—	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output or schmitt trigger input with pull-high resistors. PB0~PB7 also as A/D converter analog input pin by software programming (8 channels).
VSS	—	—	Negative power supply, GND
PC0~PC6	I/O	Pull-high or None	Bidirectional 7-bit input/output port. Software instructions determine the CMOS output or schmitt trigger input with or without pull-high resistors (mask option).

Pin Name	I/O	Mask Option	Function
$\overline{\text{RES}}$	I	—	Schmitt trigger reset input. Active low.
VDD	—	—	Positive power supply
OSC1 OSC2	I O	Crystal or RC	OSC1, OSC2 are connected to an RC network or a crystal (determined by mask option) for the internal system clock. In the case of RC operation, OSC2 is the output terminal for 1/4 system clock (NMOS open drain output)
SDA	I/O	—	Bidirectional input/output pin for H-BUS DATA line with NMOS output pin.
SCL	I/O	—	Bidirectional input/output pin for H-BUS CLOCK line with NMOS output pin.

Absolute Maximum Ratings*

Supply Voltage ....V_{SS}–0.3V to 6V	Storage Temperature .... –50°C to 125°C
Input Voltage................. V_{SS}–0.3V to V_{DD}+0.3V	Operating Temperature.............. –25°C to 70°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	—	—	4.5	5	5.5	V
I_{DD1}	Operating Current (Crystal OSC)	5V	No load, f_{SYS}= 6MHz	—	3	5	mA
I_{DD2}	Operating Current (RC OSC)	5V	No load, f_{SYS}= 6MHz	—	3	5	mA
I_{STB1}	Standby Current (WDT Enabled)	5V	No load, system HALT	—	—	10	μA
I_{STB2}	Standby Current (WDT Disabled)	5V	No load, system HALT	—	—	2	μA
V_{IL1}	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
V_{IH1}	Input High Voltage for I/O Ports	5V	—	3.5	—	5	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL2}	Input Low Voltage (TMR, $\overline{INT}$)	5V	—	0	—	1.3	V
V_{IH2}	Input High Voltage (TMR, $\overline{INT}$)	5V	—	3.8	—	5	V
V_{IL3}	Input Low Voltage ($\overline{RES}$)	5V	—	—	2.5	—	V
V_{IH3}	Input High Voltage ($\overline{RES}$)	5V	—	—	4.0	—	V
I_{OL}	I/O Port Sink Current	5V	V_{OL}=0.5V	4	8	—	mA
I_{OH}	I/O Port Source Current	5V	V_{OH}=4.5V	–2	–4	—	mA
R_{PH}	Pull-high Resistance of I/O Ports	5V	—	10	30	50	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS1}	System Clock (Crystal OSC)	5V	—	400	—	6000	kHz
f_{SYS2}	System Clock (RC OSC)	5V	—	400	—	6000	kHz
f_{TIMER}	Timer I/P Frequency (TMR)	5V	—	0	—	4000	kHz
t_{WDTOSC}	Watchdog Oscillator	5V	—	35	65	130	µs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	µs
t_{SST}	System Start-up timer Period	—	Power-up or wake-up from the halt mode	—	1024	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	µs
t_{AD1}	A/D Conversion Time	—	f_{SYS}=6MHz A/D Clock= 1/2 f_{SYS}	—	40	80	µs
t_{AD2}	A/D Conversion Time	—	f_{SYS}=6MHz A/D Clock= 1/8 f_{SYS}	—	150	300	µs
t_{AD3}	A/D Conversion Time	—	f_{SYS}=6MHz A/D Clock= 1/32 f_{SYS}	—	600	1000	µs

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

Functional Description

Execution flow

The HT46C20 system clock 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 ROM 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 one. The program counter then points to the memory word containing the next instruction code.

When executing a jump instruction, conditional skip execution, loading PCL register, subroutine call, initial reset, internal interrupt, external interrupt or return from subroutine, 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.

Program memory – ROM

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 2048×14 bits, addressed by the program counter and table pointer.

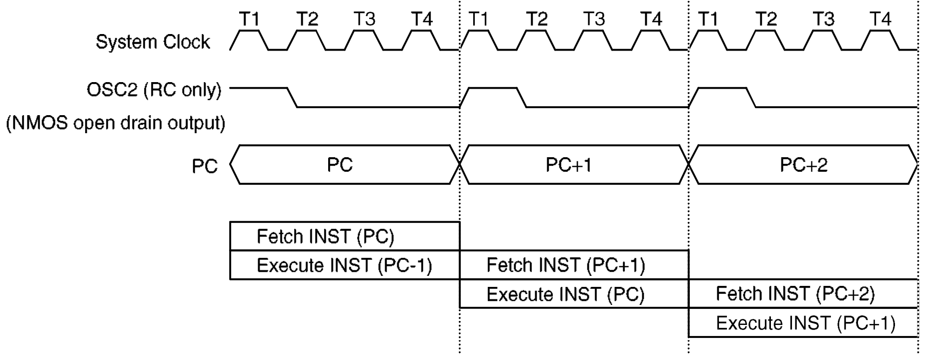
Certain locations in the program memory are reserved for special usage:

- 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 $\overline{\text{INT}}$ input pin is activated, and the interrupt is enabled and the stack is not full, the program begins execution at location 004H.



Execution flow

- Location 008H

  This area is reserved for the timer/event counter interrupt service program. If the timer interrupt resulting from a timer/event counter overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H.

- Location 00CH

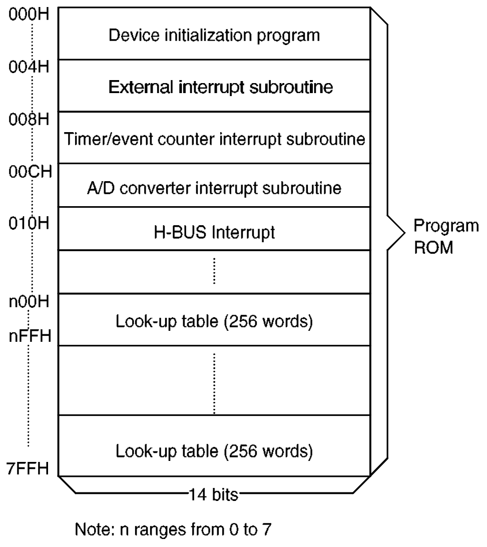
  This area is reserved for the A/D converter interrupt service program. If the A/D converter interrupt resulting from the A/D converter is completed, and if the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.

- Location 010H

  This area is reserved for the H-BUS interrupt service program. If the H-BUS interrupt resulting from a slave address is matched or completed one byte of data transfer, 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, 1 page=256 words) and TABRDL [m] (the last page) transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined, the other bits of the table word are transferred to the lower portion of TBLH, the remaining 2 bits are read as "0". The table


000H
Device initialization program
004H
External interrupt subroutine
008H
Timer/event counter interrupt subroutine
00CH
A/D converter interrupt subroutine
010H
H-BUS Interrupt
n00H
nFFH
Look-up table (256 words)
7FFH
Look-up table (256 words)
Program ROM
14 bits
Note: n ranges from 0 to 7

Program memory

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
Timer/event counter overflow	0	0	0	0	0	0	0	1	0	0	0
A/D converter interrupt	0	0	0	0	0	0	0	1	1	0	0
H-BUS interrupt	0	0	0	0	0	0	1	0	0	0	0
Skip	**PC+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

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

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 2 cycles to complete the operation. These areas may function as normal program memory depending upon the requirements.

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 six 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 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 six return addresses are stored).

Data memory – RAM

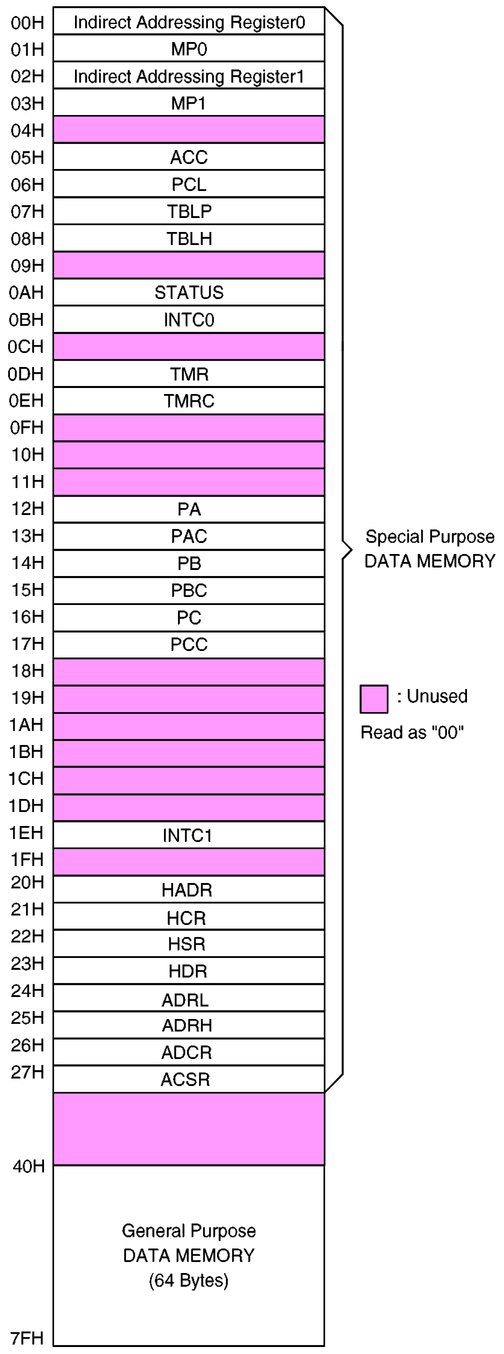
The data memory is designed with 89×8 bits. The data memory is divided into two functional groups: special function registers and general purpose data memory (64×8). Most are read/write, but some are read only.

The special function registers include the indirect addressing register 0 (00H), the indirect addressing register 1 (02H), the timer/event counter (TMR;0DH), the timer/event counter control register (TMRC;0EH), the program counter lower-order byte register (PCL;06H), the memory pointer register 0 (MP0;01H), the memory pointer register 1 (MP1;03H), the accumulator (ACC;05H), the table pointer (TBLP;07H), the table higher-order byte register (TBLH;08H), the status register (STATUS;0AH), the interrupt control register 0 (INTC0;0BH), the I/O registers (PA;12H, PB;14H, PC;16H), the I/O control registers (PAC;13H, PBC;15H, PCC;17H), the interrupt control register 1 (INTC1;1EH), the H-BUS slave address register (HADR;20H), the H-BUS control register (HCR;21H), the H-BUS status register

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

Notes: *10~*0: Table location bits
@7~@0: Table pointer bits
P10~P8: Current program counter bits



RAM mapping

(HSR;22H), the H-BUS data register (HDR;23H), the A/D result lower-order byte register (ADRL;24H), the A/D result higher-order byte register (ADRH;25H), the A/D control register (ADCR;26H) and the A/D clock source register (ACSR;27H). The remaining space before the 40H is reserved for future expanded usage and reading these locations 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 directly. 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 instructions, respectively. They are also indirectly accessible through the memory pointer register (MP0;01H, MP1;03H).

Indirect addressing register

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

The function of data movement between two indirect addressing registers, is not supported. The memory pointer registers MP0 (01H) and MP1 (03H) are 7-bit register which can be used to access the data memory by combining corresponding indirect addressing registers. The bit 7 of MP0 and of MP1 are undefined and reading will return the result “1”. Any writing operation to MP0 and MP1 will only transfer the lower 7-bit data to MP0 and MP1.

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 also changes 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 (PD) and watchdog time-out flag (TO). It also records the status information and controls the operation sequence.

With the exception of the TO and PD 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 PD flags. In addition it should be noted that operations related to the status register may give different results from those intended. The TO and PD flags can only be changed by the watchdog timer overflow, chip 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 automatically pushed onto the stack. If the contents of status are important and if the subroutine can corrupt the status register, precautions must be taken to save it properly.

Interrupt

The HT46C20 provides an external interrupt, an internal timer/event counter interrupt, the A/D converter interrupt and the H-BUS interrupts. The interrupt control register 0 (INTC0;0BH) and interrupt control register 1 (INTC1;1EH) contains the interrupt control bits to set the enable/disable and the interrupt request flags.

Labels	Bits	Function
C	0	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.
AC	1	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.
Z	2	Z is set if the result of an arithmetic or logic operation is zero; otherwise Z is cleared.
OV	3	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.
PD	4	PD is cleared when either a system power-up or executing the CLR WDT instruction. PD is set by executing the HALT instruction.
TO	5	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	Undefined, read as “0”
—	7	Undefined, read as “0”

Status 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 the INTC0 or INTC1 may be set to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the 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, followed by a branch to a subroutines at specified locations in the program memory. Only the program counter is pushed onto the stack. If the contents of the register and status register (STATUS) are altered by the interrupt service program which corrupt the desired control sequence, then the contents must be saved first.

External interrupt is triggered by a high to low transition of the $\overline{\text{INT}}$ (bit 5 of PAC must be set to “1”; PA5 is in input mode) 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 5 of INTC0), caused by a timer overflow. When the interrupt is enabled, and the stack is not full and the TF bit is set, a subroutine call to location 08H will occur. The related interrupt request flag (TF) will be reset and the EMI bit cleared to disable further interrupts.

The A/D converter interrupt is initialized by setting the A/D end of converter interrupt request flag (ADF ; bit 6 of INTC0), caused by the end of conversion of the A/D converter. When the interrupt is enabled, and the stack is not full and the ADF bit is set, a subroutine call to location 0CH will occur. The related interrupt request flag (ADF) will be reset and the EMI bit cleared to disable further interrupts.

The H-BUS interrupt is initialized by setting the H-BUS interrupt request flag (HIF; bit 4 of INTC1), caused by a slave address match (HAAS=“1”) or one byte of data transfer is completed. When the in-

Register	Bit No.	Label	Function
INTC0 (0BH)	0	EMI	Controls the master (global) interrupt (1= enabled; 0= disabled)
	1	EEI	Controls the external interrupt (1= enabled; 0= disabled)
	2	ETI	Controls the timer/event counter interrupt (1= enabled; 0= disabled)
	3	EADI	Controls the A/D converter interrupt (1= enabled; 0= disabled)
	4	EIF	External interrupt request flag (1= active; 0= inactive)
	5	TF	Internal timer/event counter request flag (1= active; 0= inactive)
	6	ADF	A/D converter request flag (1= active; 0= inactive)
	7	—	Unused bit, read as “0”

INTC0 register

terrupt is enabled, and the stack is not full and the HIF bit is set, a subroutine call to location 10H will occur. The related interrupt request flag (HIF) 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, the RET or RETI instruction may be invoked. RETI will set the EMI bit to enable an interrupt service, but RET will 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.

No.	Interrupt Source	Priority	Vector
a	External interrupt	1	04H
b	Timer/event counter overflow	2	08H
c	A/D converter interrupt	3	0CH
d	H-BUS interrupt	4	10H

The timer/event counter interrupt request flag (TF), external interrupt request flag (EIF), A/D converter interrupt request flag (ADF), the H-BUS interrupt request flag (HIF), enable timer/event counter bit (ETI), enable external interrupt bit (EEI), enable end of A/D conversion interrupt bit (EADI), enable H-BUS interrupt bit (EHI) and enable master interrupt bit (EMI) constitute an interrupt control register 0 (INTC0) and an interrupt control register 1 (INTC1) which are located at 0BH and 1EH in the data memory. EMI, EEI, ETI, EADI, EHI are used to control the enabling/disabling of interrupts. These bits prevent the requested interrupt being serviced. Once the interrupt request flags (TF, EIF, ADF, HIF) are set, they will remain in the INTC0 and INTC1 registers until the interrupts are serviced or cleared by a software instruction.

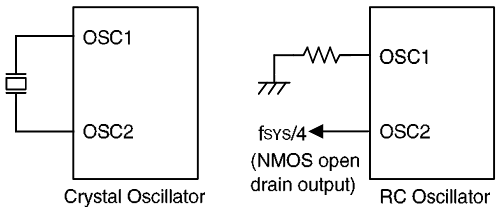
It is recommended that a program does not use the “CALL subroutine” within the interrupt subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications, if only one stack is left and enabling the interrupt is not well controlled, once the “CALL subroutine”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

Register	Bit No.	Label	Function
INTC1 (1EH)	0	EHI	Controls the H-BUS interrupt (1= enabled; 0= disabled)
	1	—	Unused bit, is read as “0”
	2	—	Unused bit, is read as “0”
	3	—	Unused bit, is read as “0”
	4	HIF	H-BUS interrupt request flag (1= active; 0= inactive)
	5	—	Unused bit, read as “0”
	6	—	Unused bit, read as “0"
	7	—	Unused bit, read as “0”

INTC1 register

Oscillator configuration

There are two oscillator circuits in the HT46C20.



System oscillator

Both are designed for system clocks; the RC oscillator and the crystal oscillator, which are determined by mask options. No matter what oscillator type is selected, the signal provides the system clock. The halt mode stops the system oscillator and ignores an external signal to conserve power.

If an RC oscillator is used, an external resistor between OSC1 and GND is needed and the resistance must range from 51kΩ to 1MΩ. The system clock, divided by 4, is available on OSC2, 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.

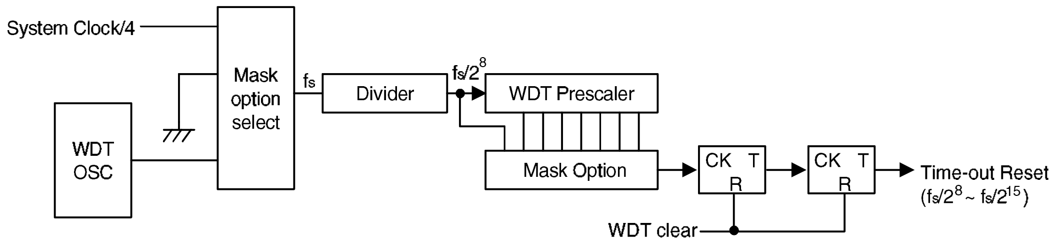
If a crystal oscillator is used, a crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift needed for oscillator, no other external components are needed. Instead of a crystal, a resonator can also be connected between OSC1 and OSC2 to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required.

The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Even if the system enters the power down mode, the system clock is stopped, but the WDT oscillator still works with a period of approximately 65 µs. The WDT oscillator can be disabled by mask option to conserve power.

Watchdog timer - WDT

The clock source of the WDT is implemented by an dedicated RC oscillator (WDT oscillator) or instruction clock (system clock divided by 4) decided by mask options. This 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 a mask option. If the watchdog timer is disabled, all the executions related to the WDT result in no operation.

Once an internal WDT oscillator (RC oscillator with period 65µs normally) is selected, it is divided by 2^{n+8} (by mask option to get the WDT time-out period). The n is from 0 to 7. The minimum period of WDT time-out period is about 17ms. This time-out period may vary with temperature, VDD and process variations. By selection the WDT mask option, longer time-out periods can be realized. If the n is selected as 7, the maximum time-out period is divided by 2^{15} about 2.1s.



Watchdog timer

If the WDT oscillator is disabled, the WDT clock may still cone from the instruction clock and operate in the same manner except that in the halt state the WDT may stop counting and lose its protecting purpose. In this situation the logic can only be restarted by external logic. If the device operates in a noisy environment, using the on-chip RC oscillator (WDT OSC) is strongly recommended, since the HALT will stop 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 PC and SP are reset to zero. To clear the contents of WDT, three methods are adopted; external reset (a low level to $\overline{\text{RES}}$), software instructions, or a HALT instruction. The software instructions 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 mask option — "CLR WDT times selection option". If the "CLR WDT" is selected (i.e. CLRWDT 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. CLRWDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip because of time-out.

If the WDT time-out period is selected $f_S/2^8$ (mask option), the WDT time-out period ranges from $f_S/2^8$~$f_S/2^9$, since the "CLR WDT" or "CLR WDT1" and "CLR WDT2" instructions only clear the last two stages of the WDT.

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 keeps running (if the WDT oscillator is selected).
- The contents of the on–chip RAM and registers remain unchanged.
- WDT will be cleared and recount again (if the WDT clock has come from the WDT oscillator).
- All I/O ports maintain their original status.
- The PD flag is set and the TO flag is cleared.

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 WDT overflow. An external reset causes a device initialization and the WDT overflow performs a "warm reset". Examining the TO and PD flags, the reason for chip reset can be determined. The PD flag is cleared when system power-up or executing the CLR WDT instruction and is set when the HALT instruction is executed. The TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the PC 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 mask 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.

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

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

Reset

There are three ways in which a reset can occur:

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

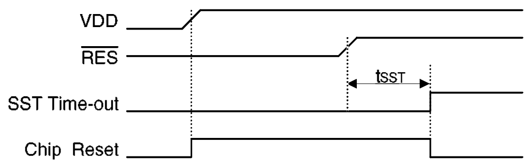
The WDT time-out during halt mode is different from other chip reset conditions, since it can perform a "warm reset" that just resets PC and SP, leaving the other circuits in their original state. Some registers remain unchanged during other reset conditions. Most registers are reset to the "initial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

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

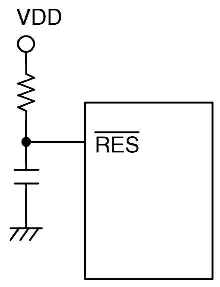
Note: "u" means "unchanged"

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

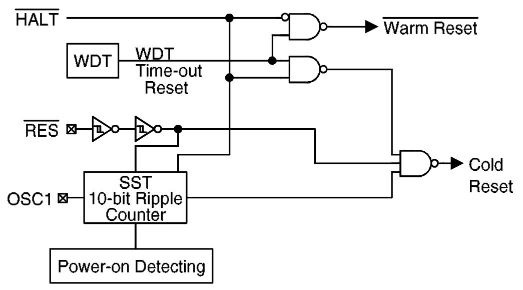
When a system power-up occurs, the SST delay is added during the reset period. But when the reset comes from the $\overline{RES}$ pin, the SST delay is disabled. Any wake-up from halt will enable the SST delay.



Reset timing chart



Reset circuit



Reset configuration

The functional unit chip reset status are shown below.

PC	000H
Interrupt	Disable
Prescaler	Clear
WDT	Clear. After master reset, WDT begins counting
Timer/event counter	Off
Input/output ports	Input mode
SP	Points to the top of the stack

The state of the registers is 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)
TMR	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMRC	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
PC	000H	000H	000H	000H	000H*
MP0	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
MP1	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu 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
PA	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PAC	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	-111 1111	-111 1111	-111 1111	-111 1111	-uuu uuuu
PCC	-111 1111	-111 1111	-111 1111	-111 1111	-uuu uuuu
HADR	0000 000-	uuuu uuu-	uuuu uuu-	uuuu uuu-	uuuu uuu-
HCR	0--0 0----	0--0 0---	0--0 0---	0--0 0---	u--u u---
HSR	100- -0-1	100- -0-1	100- -0-1	100- -0-1	uuu- -u-u
HDR	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ADRL	xx-- ----	uu-- ----	uu-- ----	uu-- ----	uu-- ----
ADRH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- --00	1--- --00	1--- --00	1--- --00	u--- --uu

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

Timer/event counter

One timer/event counter is implemented in the HT46C20. The timer/event counter contains an 8-bit programmable count-up counter and the clock may come from an external source or the prescaler output clock.

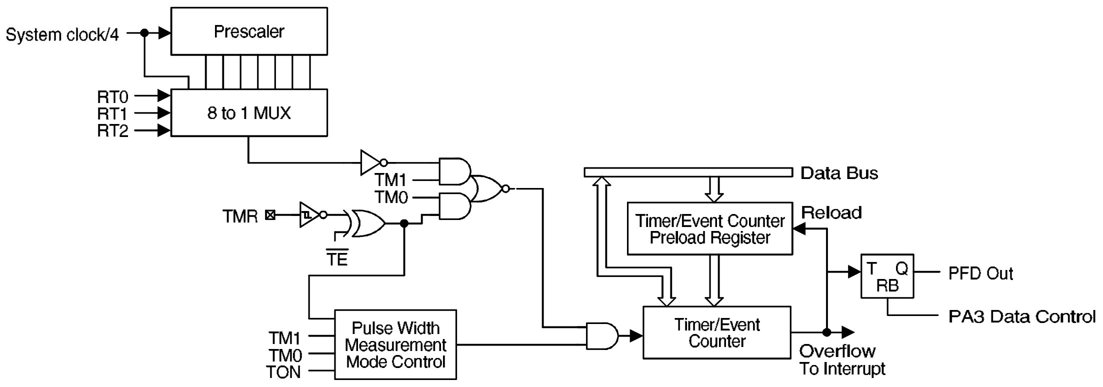
Using the internal prescaler output clock, there are eight reference time-base. The external clock input allows the user to count external events, measure time intervals or pulse widths, or to generate an accurate time base.

There are two registers related to the timer/event counter; TMR ([0DH]), TMRC ([0EH]). Two physical registers are mapped to TMR location; writing TMR makes the starting value put in the timer/event counter preload register and reading TMR gets the content of the timer/event counter. The TMRC is a timer/event counter control register, which defines some options.

The TM0, TM1 bits define the operating mode. The event count mode is used to count external events, which means the clock source comes from an external (TMR) pin. The timer mode functions as a normal timer with the clock source coming from the timer prescaler output clock. 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 prescaler output clock.

Label (TMRC)	Bits	Function
RT0 RT1 RT2	0 1 2	Defines the prescaler option
TE	3	To define the TMR active edge of timer/event counter (0= active on low to high; 1= active on high to low)
TON	4	To enable/disable timer counting (0= disabled; 1= enabled)
—	5	Unused bits, read as “0”
TM0 TM1	6 7	To define the operating mode 01= Event count mode (external clock) 10= Timer mode (internal clock) 11= Pulse width measurement mode 00= Unused

TMRC register



Timer/Event Counter

In the event count or timer mode, once the timer/event counter starts counting, it will count from the current contents in the timer/event counter to FFH. Once overflow occurs, the counter is reloaded from the timer/event counter preload register and generates the interrupt request flag (TF; bit 5 of INTC) at the same time.

In pulse width measurement mode with the TON and TE bits are equal to one, 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 measurements 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 operating 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 the interrupt request just like the other two modes.

The RT0, RT1, RT2 bits define the prescaler option. There are eight kinds of option, which is used to select the internal timer clock source is divided by 2^n, (n from 0 to 7).

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 cleared automatically after the measurement cycle is complete. But in the other two modes the TON can only be reset by instruction. The overflow of the timer/event counter is one of the wake-up sources. No matter what the operation mode is, writing a 0 to ETI can disable the interrupt service.

In the case of timer/event counter OFF condition, writing data to the timer/event counter preload register will also reload that data to timer/event counter. But if the timer/event counter is turned on, data written to the timer/event counter will only be kept in the timer/event counter preload register. The timer/event counter will still operate until the overflow occurs.

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

In the event count or pulse width measurement mode, the PA4 must set to "1" (bit 4 of the PA pin in input mode).

The timer prescaler input is system clock/4 and the output of the timer prescaler is decided by RT2, RT1 and RT0.

RT2	RT1	RT0	Division Ratio
0	0	0	1:1
0	0	1	1:2
0	1	0	1:4
0	1	1	1:8
1	0	0	1:16
1	0	1	1:32
1	1	0	1:64
1	1	1	1:128

Timer prescaler option

Input/output ports

There are 21 bidirectional input/output lines in the HT46C20, labeled from PA to PC, which are mapped to the data memory of [12H], [14H] and [16H] respectively. All these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction MOV A,[m] (m=12H, 14H or 16H). For output operation, all data is latched and remains unchanged until the output latch is rewritten.

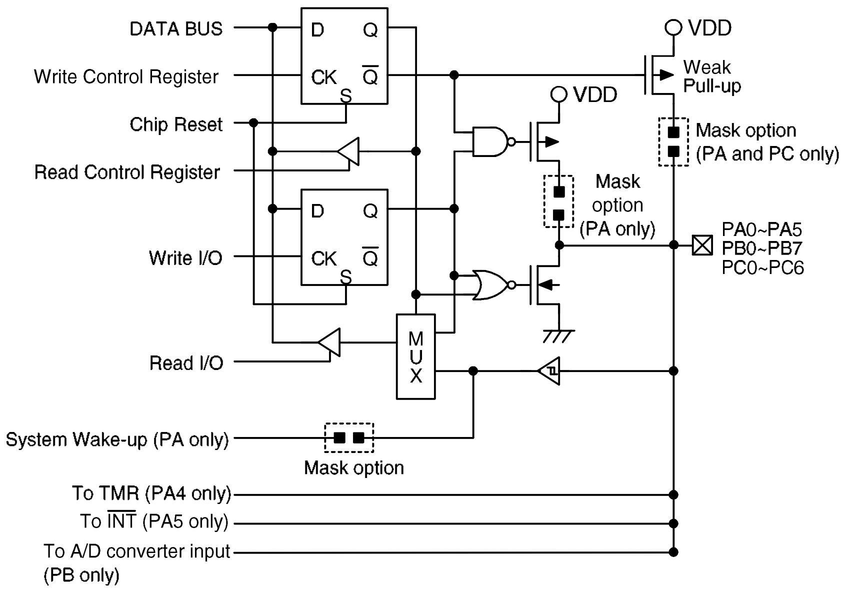
Each I/O line has it's own control register (PAC, PBC, PCC) to control the input/output configuration. With this control register, CMOS (PA CMOS or NMOS by mask option, PB and PC CMOS) output or schmitt trigger input with or without pull-high resistors (PA and PC by mask option, PB with pull-high resistor) structures can be reconfigured dynamically (i.e., on-the-fly) under software control. To function as an input, the corresponding latch of the control register must write "1". The pull-high resistance will exhibit automatically if the pull-high option is selected. The input source also depends on the control register. If the control register bit is "1", input will read the pad state. If the control register bit is "0", the contents of the latches will move to the internal bus. The latter is possible in "read-modify-write" instruction. For output function, CMOS or NMOS is the configuration by mask option (PA only). These control registers are mapped to locations 13H, 15H and 17H.

After a chip reset, these input/output lines remain at high levels or floating (mask option). Each bit of these input/output latches can be set or cleared by the SET [m].i or CLR [m].i (m=12H, 14H or 16H) instruction.

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

Each line of port A (PA0 to PA5) has the capability to wake-up the device. The highest one bit of port C is not physically implemented, on reading it a "0" is returned and writing results in a no-operation. On reading the highest two bits of port A are unknown.

The PA0 and PA1 are buzzer output by mask option. The PA2 is 38kHz or 40kHz frequency output (only 455kHz or 480kHz crystal oscillation) by mask option. The PA3 is PFD output by mask option. The PA4 and PA5 are connected to TMR and $\overline{\text{INT}}$ pad, when there's an external clock and signal input to TMR and $\overline{\text{INT}}$ pad, the PA4 and PA5 must be set as input pin. PB port configuration can be an analog input or normal I/O port (CMOS output or input with pull-high resistor) by ADCR register.



Input/output ports

HOLTEK-BUS Serial Interface

HOLTEK-BUS (in short: H-BUS) is implemented in the HT46C20. The H-BUS is a bidirectional two-wire lines. The data line and clock line are implement in SDA pin and SCL pin. The SDA and SCL are NMOS open drain output pin. They must connect a pull-high resistor respectively.

Using the H-BUS, the HT46C20 has two ways to transfer data. One is in slave transmit mode, the other is in slave receive mode. There are four registers related to H-BUS; HADR([20H]), HCR([21H]), HSR([22H]), HDR([23H]). The HADR register is the slave address setting of the HT46C20, if the master sends the calling address which match, it means that this device is selected. The HCR is H-BUS control register which defines the HT46C20 enable or disable the H-BUS as a transmitter or as a receiver. The HSR is H-BUS status register, it responds with the H-BUS status. The HDR is input/output data register, data to transmit or receive must be via the HDR register.

The H-BUS control register contains three bits. The HEN bit define the enable or disable the H-BUS. If the data wants transfer via H-BUS, this bit must be set. The HTX bit defines whether the H-BUS is in transmit or receive mode. If the device is as a transmitter, this bit must be set to “1”. The TXAK defines the transmit acknowledge signal, when the HT46C20 received 8-bit data, the HT46C20 sends this bit to H-BUS at the 9th clock. If the receiver wants to continue to receive the next data, this bit must be reset to “0” before receiving data.

The H-BUS status register contains 5 bits. The HCF bit is reset to “0” when one data byte is being transferred. If one data transfer is completed, this bit is set to “1”. The HASS bit is set “1” when the address is match, and the H-BUS interrupt request flag is set to “1”. If the interrupt is enabled and the stack is not full, a subroutine call to location 10H will occur. Writing data to the H-BUS control register clears HAAS bit. If the address is not match, this bit is reset to “0”. The HBB bit is set to respond the H-BUS is busy. It mean that a START signal is detected. This bit is reset to “0” when the H-BUS is not busy. It means that a STOP signal is detected and the H-BUS is free. The SRW bit defines the read/write command bit, if the calling address is match. When HAAS is set to “1”, the HT46C20 check SRW bit to determine whether the HT46C20 is working in transmit or receive mode. When SRW bit is set “1”, it means that the master wants to read data from H-BUS, the slave device must write data to H-BUS, so the slave device is working in transmit mode. When SRW is reset to “0”, it means that the master wants to write data to H-BUS, the slave device must read data from the bus, so the slave device is working in receive mode. The RXAK bit is reset “0” indicates an acknowledges signal has been received. In the transmit mode, the transmitter checks RXAK bit to know the receiver which wants to receive the next

Label(HCR)	Bits	Function
HEN	7	To enable/disable H-BUS function (0= disable; 1= enable)
—	6	Unused bits, read as “0”
—	5	Unused bits, read as “0”
HTX	4	To define the transmit/receive mode (0= receive mode; 1= transmit)
TXAK	3	To enable/disable transmit acknowledge (0= acknowledge; 1= don’t acknowledge)
—	0~2	Unused bits, read as “0”

HCR register

data byte, so the transmitter continue to write data to the H-BUS until the RXAK bit is set to "1" and the transmitter releases the SDA line, so that the master can send the STOP signal to release the bus.

The HADR bit7-bit1 define the device slave address. At the beginning of transfer, the master must select a device by sending the address of the slave device. The bit 0 is unused and is not defined. If the H-BUS receives a start signal, all slave device notice the continuity of the 8-bit data. The front of 7 bits is slave address and the first bit is MSB. If the address is match, the HAAS status bit is set and generate an H-BUS interrupt. In the ISR, the slave device must check the HAAS bit to know the H-BUS interrupt comes from the slave address that has match or completed one 8-bit data transfer. The last bit of the 8-bit data is read/write command bit, it responds in SRW bit. The slave will check the SRW bit to know if the master wants to transmit or receive data. The HT46C20 check SRW bit to know it is as a transmitter or receiver.

Bit7~Bit1	Bit0
Slave Address	—

HADR register

"–" means undefined

The HDR register is the H-BUS input/output data register. Before transmitting data, the HDR must write the data which we want to transmit. Before receiving data, the HT46C20 must dummy read data from HDR. Transmit or Receive data from H-BUS must be via the HDR register. At the beginning of the transfer of the H-BUS, the HT46C20 must initial the bus, the following are the notes for initialing the H-BUS:

1: Write the H-BUS address register (HADR) to define its own slave address.

2: Set HEN bit of H-BUS control register (HCR) bit 0 to enable the H-BUS.

3: Set EHI bit of the interrupt control register 1 (INTC1) bit 0 to enable the H-BUS interrupt.

Label(HSR)	Bits	Function
HCF	7	HCF is clear to "0" when one data byte is being transferred, HCF is set to "1" indicating 8-bit data communication has been finished.
HAAS	6	HAAS is set to "1" when the calling addressed is matched, and H-BUS interrupt will occur and HIF is set.
HBB	5	HBB is set to "1" when H-BUS is busy and HBB is cleared to "0" means that the H-BUS is not busy.
—	4	Unused bit
—	3	Unused bit
SRW	2	SRW is set to "1" when the master wants to read data from the H-BUS, so the slave must transmit data to the master. SRW is cleared to "0" when the master wants to write data to the H-BUS, so the slave must receive data from the master.
—	1	Unused bit
RXAK	0	RXAK is cleared to "0" when the master receives an 8-bit data and acknowledgment at the 9th clock, RXAK is set to "1" means not acknowledged.

HSR register

Start signal

The START signal is generated only by the master device. The other device in the bus must detect the START signal to set the H-BUS busy bit (HBB). The START signal is SDA line from high to low, when SCL is high.

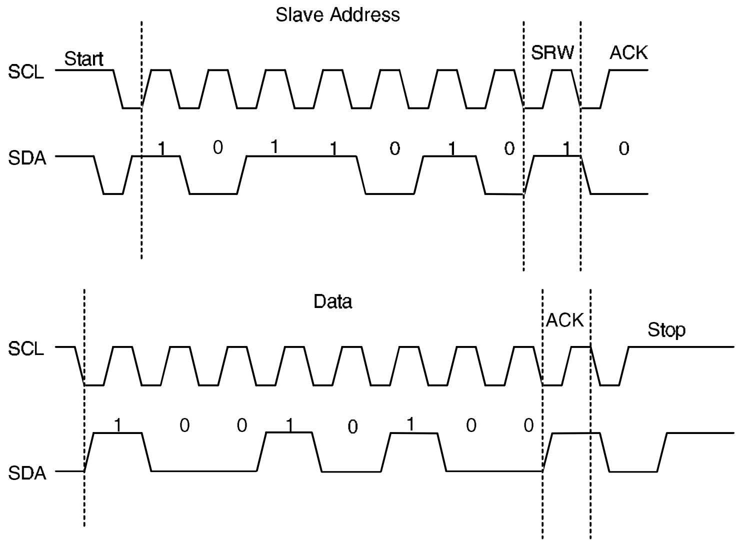
Slave address

The master must select a device for transferring the data by sending the slave device address after the START signal. All device in the H-BUS will receive the H-BUS slave address (7 bits) to compare with its own slave address (7 bits). If the slave address is matched, the slave device will generate an interrupt and save the following bit (8th bit) to SRW bit and sends an acknowledge bit (low level) to the 9th bit. The slave device also sets the status flag (HAAS), when the slave address is matched.

In interrupt subroutine, check HAAS bit to know whether the H-BUS interrupt comes from a slave address that is matched or a data byte transfer is completed. When the slave address is matched, the HT46C20 must be in transmit mode or receive mode and write data to HDR or dummy read from HDR to release the SCL line.

SRW bit

The SRW bit means that the master device wants to read from or write to the H-BUS. The slave device check this bit to understand itself if it is a transmitter or a receiver. The SRW bit is set to “1” means that the master wants to read data from the H-BUS, so the slave device must write data to a bus as a transmitter. The SRW is cleared to “0” means that the master wants to write data to the H-BUS, so the slave



S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send aclcnowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter; TXAK bit for receiver 1 bit)
P=Stop (1bit)

S	SA	SR	M	D	A	D	A	------	S	SA	SR	M	D	A	D	A	------	P

device must read data from the H-BUS as a receiver.

Acknowledge bit

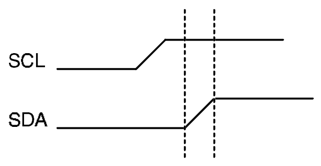
One of the slave device generates an acknowledge signal, when the slave address is matched. The master device can check this acknowledge bit to know if the slave device accepts the calling address. If no acknowledge bit, the master must send a STOP bit and end the communication. When the H-BUS status register bit 6 HAAS is high, it means the address is matched, so the slave must check SRW as a transmitter (set HTX) to “1” or as a receiver (clear HTX) to “0”.

Data byte

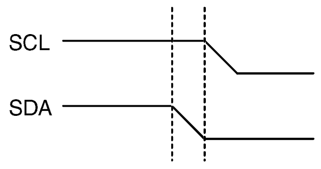
The data is 8 bits and is sent after the slave device has acknowledges the slave address. The first bit is MSB and the 8th bit is LSB. The receiver sends the acknowledge signal (“0”) and continues to receive the next one byte data. If the transmitter checks and there’s no acknowledge signal, then it release the SDA line, and the master sends a STOP signal to release the H-BUS. The data is stored in the HDR register. The transmitter must write data to the HDR before transmit data and the receiver must read data from the HDR after receiving data.

Receive acknowledge bit

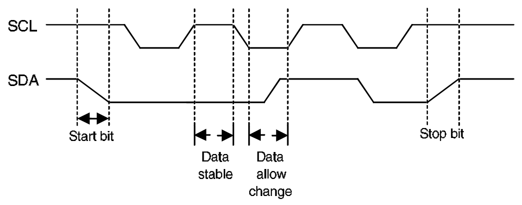
When the receiver wants to continue to receive the next data byte, it generates an acknowledge bit (TXAK) at the 9th clock. The transmitter checks the acknowledge bit (RXAK) to continue to write data to the H-BUS or change to receive mode and dummy read the HDR register to release the SDA line and the master sends the STOP signal.



Stop bit

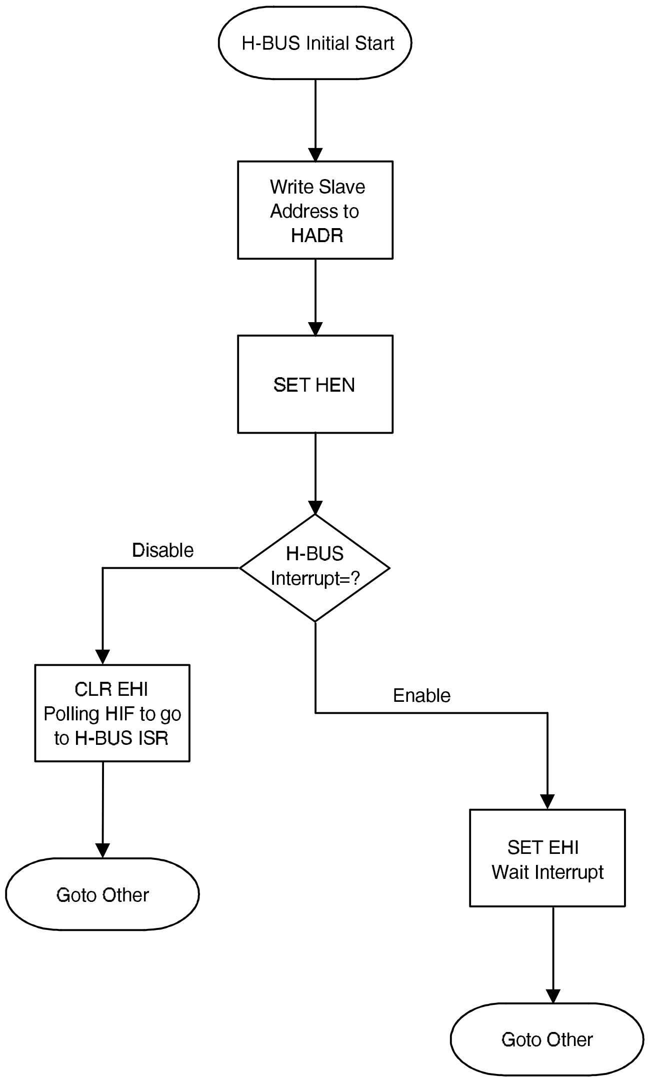


Start bit

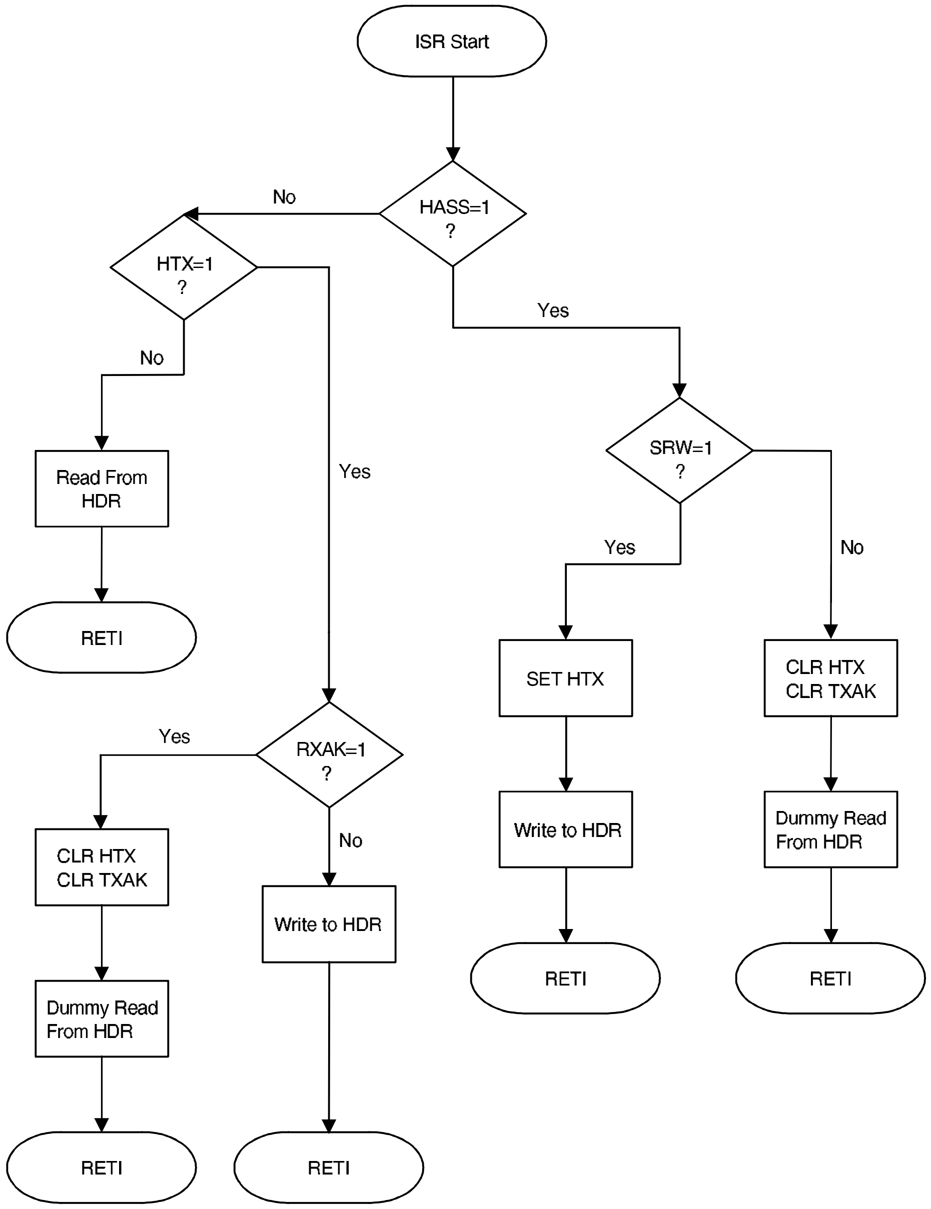


Data stable and data allow change

The H-BUS initial program flow chart as follows:



The H-BUS ISR program flow chart as follows:



A/D converter

The 8 channels and 10-bit resolution A/D Converter is implemented in the HT46C20. The reference voltage is VDD. The A/D Converter contains four special registers which are; ADRL([24H]), ADRH([25H]), ADCR([26H]) and ACSR([27H]). The ADRH and ADRL are A/D result register higher-order byte and lower-order byte. After the A/D conversion is completed, the ADRH and ADRL are read to get the conversion result data and clear the $\overline{\text{EOC}}$ flag. The ADCR is A/D converter control register, which defines the A/D channel number, analog channel select, start A/D conversion control bit and the end of A/D conversion flag. If the HT46C20 wants to start A/D conversion, select the analog channel, and give START bit a falling edge. At the end of conversion, the $\overline{\text{EOC}}$ bit is cleared and an A/D converter interrupt occurs. The ACSR is A/D clock source register, which is used to select the A/D clock source.

Bit7	**Bit6**	**Bit5~Bit0**
D1	D0	Unused bits

ADRL register

The A/D converter control register is used to control the A/D converter. The bit2-bit0 of ADCR is used to select an analog input channel. There are a total of eight channels to select. The bit5-bit3 of ADCR is used to set port B configuration. Port B can be an analog input or as digital input port by this three bits. The $\overline{\text{EOC}}$ bit (A/D Control Register bit 6) is end of A/D conversion flag. Check this bit to know when A/D conversion is completed. The START bit of the A/D control register is used to define the A/D conversion has began. Give START bit a falling edge that means the A/D conversion has started.

The bit 7 of A/D clock source register is used for test mode. The bits 1 and 0 are selected as A/D converter clock source.

When conversion is completed, the A/D interrupt request flag is set. At the A/D interrupt service subroutine, the $\overline{\text{EOC}}$ bit is set when read data from ADRH and ADRL.

Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
D9	D8	D7	D6	D5	D4	D3	D2

ADRH register

Label (ACSR)	**Bits**	**Function**
ADCS0 ADCS1	0 1	Selects the A/D converter clock source 00= system clock÷2 01= system clock÷8 10= system clock÷32 11= undefined
—	2~6	Unused bits, read as “0”
TEST	7	For test mode used only

ACSR register

Label (ADCR)	Bits	Function
ACS0 ACS1 ACS2	0 1 2	Defines the analog channel select.
PCR0 PCR1 PCR2	3 4 5	Defines port B configuration select.
EOC	6	Gives response at the end of the A/D conversion. (0= end of A/D conversion)
START	7	Starts the A/D conversion. (1→ 0= start; 1→ reset A/D converter)

ADCR register

PCR2	PCR1	PCR0	7	6	5	4	3	2	1	0
0	0	0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0	0	1	PB7	PB6	PB5	PB4	PB3	PB2	PB1	A0
0	1	0	PB7	PB6	PB5	PB4	PB3	PB2	A1	A0
0	1	1	PB7	PB6	PB5	PB4	PB3	A2	A1	A0
1	0	0	PB7	PB6	PB5	PB4	A3	A2	A1	A0
1	0	1	PB7	PB6	PB5	A4	A3	A2	A1	A0
1	1	0	PB7	PB6	A5	A4	A3	A2	A1	A0
1	1	1	A7	A6	A5	A4	A3	A2	A1	A0

Port B configuration

ACS2	ACS1	ACS0	Analog Channel
0	0	0	A0
0	0	1	A1
0	1	0	A2
0	1	1	A3
1	0	0	A4
1	0	1	A5
1	1	0	A6
1	1	1	A7

Analog input channel selection

Buzzer

The PA0 and PA1 can be used as a buzzer output pair by mask option. The differential output pair signal frequency is the WDT clock source divided by 2^{n+2}, where n is from 0 to 7 (mask option). If the PA0 and PA1 are configured as buzzer output, writing "1" to the PA0 and PA1 will enable the buzzer outputs and writing "0" to the PA0 and PA1 will disable the buzzer output. Enable by setting PA0 and PA1 to 1. Clearing PA0 and PA1 will force PA0 and PA1 to a low level and disable the buzzer output as well.

Carrier frequency

The PA2 can be configured as an output with or without a carrier driving capacity by mask option. In other words, it is easy to interface with an infrared diode. The carrier output is enabled by writing "1" to the PA2 and writing "0" to the PA2 will force PA2 to a low level and disable the carrier output as well. The IR carrier frequency is system clock divided by 12.

PFD

The HT46C20 contains a PFD for the frequency setting of the output tone. The PA3 can be used as a PFD output port by mask option. The PFD is made up of an 8-stage timer/event counter. The PFD is enabled by enabling the timer mode and writing data to TMR to change the PFD output frequency. The PFD output is enabled by writing '1' to the PA3 and writing "0" to the PA3 will force PA3 to a low level and disable the PFD output as well. If the timer is off the PA3 output is always in low level.

Frequency of PFD

$$\text{PA3}=\frac{\text{Prescaler output frequency}}{2\times(256\text{-the content of the TMR0})}$$

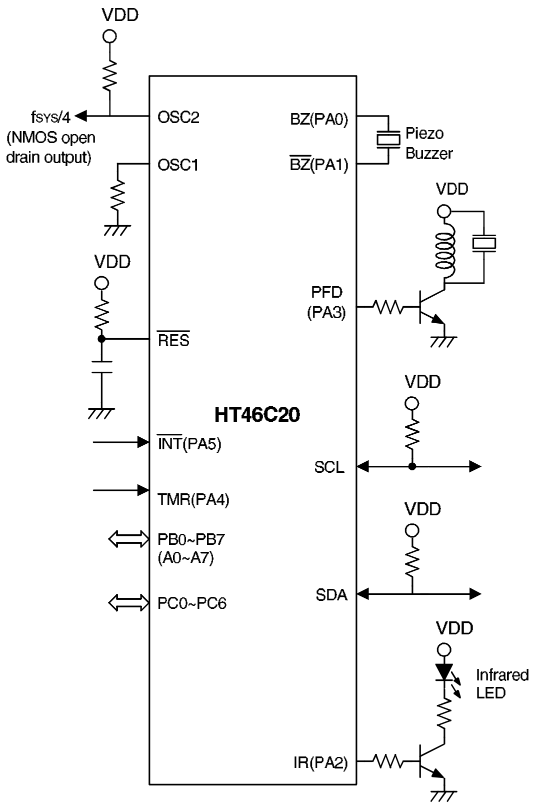
Mask option

The following shows 11 kinds of mask option in the HT46C20. ALL the mask options must be defined to ensure proper system function.

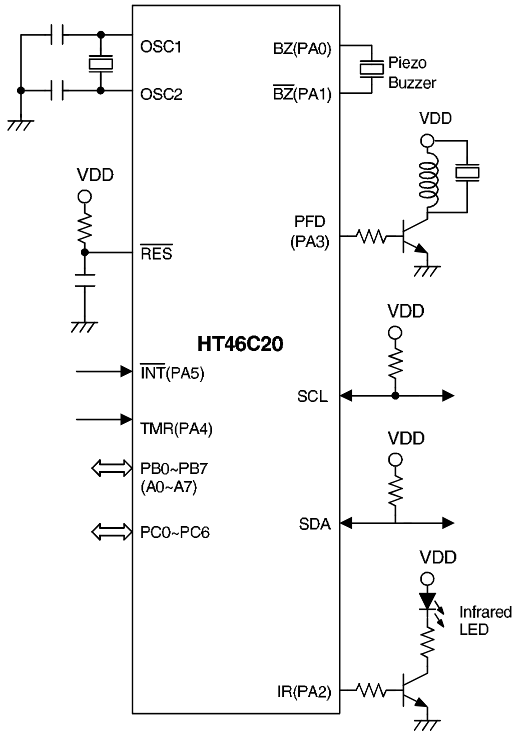
No.	Mask Option
1	OSC type selection. This option is to decide if an RC or crystal oscillator is chosen as system clock.
2	WDT source selection. There are three types of selection: on-chip RC oscillator, instruction clock or disable the WDT.
3	CLRWDT times selection. This option defines how to clear the WDT by instruction. “One time” means that the CLR WDT instruction can clear the WDT. “Two times” means only if both of the CLR WDT1 and CLR WDT2 instructions have been executed, then WDT can be cleared.
4	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.
5	Pull-high selection. This option is to decide whether a pull-high resistance is visible or not in the input mode of the I/O ports. PA and PC can be independently selected.
6	CMOS/NMOS selection. This option is to decide whether CMOS or NMOS in the output mode of the I/O ports. Each bit of PA can be independently selected.
7	WDT time-out period selection. There are eight types of selection: $F \div 2^{n+8}$, n from 0 to 7. F is WDT clock source selection.
8	BUZZER selection: Enable or disable the buzzer output.
9	BUZZER frequency selection: There are eight types of selection: $F \div 2^{n+2}$, where n is from 0 to 7. F is WDT clock source selection.
10	38kHz or 40kHz carrier frequency selection: Enable or disable the 38kHz or 40kHz carrier frequency output.
11	PFD selection: Enable or disable the PFD frequency output.

Application Circuits

RC oscillator for multiple I/O applications



Crystal oscillator or ceramic resonator for multiple I/O applications



Instruction Set Summary

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

Mnemonic	Description	Flag Affected
Data Move		
MOV A,[m]	Move data memory to ACC	None
MOV [m],A	Move ACC to data memory	None
MOV A,x	Move immediate data to ACC	None
Bit Operation		
CLR [m].i	Clear bit of data memory	None
SET [m].i	Set bit of data memory	None
Branch		
JMP addr	Jump unconditional	None
SZ [m]	Skip if data memory is zero	None
SZA [m]	Skip if data memory is zero with data movement to ACC	None
SZ [m].i	Skip if bit i of data memory is zero	None
SNZ [m].i	Skip if bit i of data memory is not zero	None
SIZ [m]	Skip if increment data memory is zero	None
SDZ [m]	Skip if decrement data memory is zero	None
SIZA [m]	Skip if increment data memory is zero with result in ACC	None
SDZA [m]	Skip if decrement data memory is zero with result in ACC	None
CALL addr	Subroutine call	None
RET	Return from subroutine	None
RET A,x	Return from subroutine and load immediate data to ACC	None
RETI	Return from interrupt	None
Table Read		
TABRDC [m]	Read ROM code (current page) to data memory and TBLH	None
TABRDL [m]	Read ROM code (last page) to data memory and TBLH	None
Miscellaneous		
NOP	No operation	None
CLR [m]	Clear data memory	None
SET [m]	Set data memory	None
CLR WDT	Clear watchdog timer	TO,PD
CLR WDT1	Pre-clear watchdog timer	TO*,PD*
CLR WDT2	Pre-clear watchdog timer	TO*,PD*
SWAP [m]	Swap nibbles of data memory	None
SWAPA [m]	Swap nibbles of data memory with result in ACC	None
HALT	Enter power down mode	TO,PD

Notes:

x: 8 bits immediate data
m: 7 bits data memory address
A: accumulator
i: 0~7 number of bits
addr: 11 bits program memory address

√: Flag is affected
–: Flag is not affected
*: Flag may be affected by the execution status

Instruction Definition

ADC A,[m] Add data memory and carry to the accumulator

Description The contents of the specified data memory, accumulator and the carry flag are added simultaneously, leaving the result in the accumulator.

Operation ACC ← ACC+[m]+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADCM A,[m] Add the accumulator and carry to data memory

Description The contents of the specified data memory, accumulator and the carry flag are added simultaneously, leaving the result in the specified data memory.

Operation [m] ← ACC+[m]+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADD A,[m] Add data memory to the accumulator

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)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADD A,x Add immediate data to the accumulator

Description The contents of the accumulator and the specified data are added, leaving the result in the accumulator.

Operation ACC ← ACC+x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADDM A,[m] Add the accumulator to the data memory

Description The contents of the specified data memory and the accumulator are added. The result is stored in the data memory.

Operation [m] ← ACC+[m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

AND A,[m] Logical AND accumulator with data memory

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)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

AND A,x Logical AND immediate data to the accumulator

Description Data in the accumulator and the specified data perform a bitwise logical_AND operation. The result is stored in the accumulator.

Operation ACC ← ACC “AND” x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

ANDM A,[m] Logical AND data memory with the accumulator

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)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

CALL addr Subroutine call

Description The instruction unconditionally calls a subroutine located at the indicated address. The program counter increments once to obtain the address of the next instruction, and pushes this onto the stack. The indicated address is then loaded. Program execution continues with the instruction at this address.

Operation Stack ← PC+1
PC ← addr

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR [m] Clear data memory

Description The contents of the specified data memory are cleared to zero.

Operation [m] ← 00H

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR [m].i Clear bit of data memory

Description The bit i of the specified data memory is cleared to zero.

Operation [m].i ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR WDT Clear watchdog timer

Description The WDT and the WDT Prescaler are cleared (re-counting from zero). The power down bit (PD) and time-out bit (TO) are cleared.

Operation WDT and WDT Prescaler ← 00H
PD and TO ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0	0	–	–	–	–

CLR WDT1 Preclear watchdog timer

Description The TD, PD flags, WDT and the WDT Prescaler has cleared (re-counting from zero), if the other preclear WDT instruction has been executed. Only execution of this instruction without the other preclear instruction sets the indicated flag which implies that this instruction has been executed and the TO and PD flags remain unchanged.

Operation WDT and WDT Prescaler ← 00H*
PD and TO ← 0*

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0*	0*	–	–	–	–

CLR WDT2 Preclear watchdog timer

Description The TO, PD flags, WDT and the WDT Prescaler are cleared (re-counting from zero), if the other preclear WDT instruction has been executed. Only execution of this instruction without the other preclear instruction sets the indicated flag which implies that this instruction has been executed and the TO and PD flags remain unchanged.

Operation WDT and WDT Prescaler ← 00H*
PD and TO ← 0*

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0*	0*	–	–	–	–

CPL [m] Complement data memory

Description Each bit of the specified data memory is logically complemented (1's complement). Bits which previously contained a one are changed to zero and vice-versa.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

CPLA [m] Complement data memory and place result in the accumulator

Description: Each bit of the specified data memory is logically complemented (1's complement). Bits which previously contained a one are changed to zero and vice-versa. The complemented result is stored in the accumulator and the contents of the data memory remain unchanged.

Operation: ACC ← $[\overline{m}]$

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

DAA [m] Decimal-Adjust accumulator for addition

Description: The accumulator value is adjusted to the BCD (Binary Code Decimal) code. The accumulator is divided into two nibbles. Each nibble is adjusted to the BCD code and an internal carry (AC1) will be done if the low nibble of the accumulator is greater than 9. The BCD adjustment is done by adding 6 to the original value if the original value is greater than 9 or a carry (AC or C) is set; otherwise the original value remains unchanged. The result is stored in the data memory and only the carry flag (C) may be affected.

Operation:
If ACC.3~ACC.0 >9 or AC=1
then [m].3~[m].0 ← (ACC.3~ACC.0)+6, AC1=$\overline{AC}$
else [m].3~[m].0) ← (ACC.3~ACC.0), AC1=0
and
If ACC.7~ACC.4+AC1 >9 or C=1
then [m].7~[m].4 ← ACC.7~ACC.4+6+AC1,C=1
else [m].7~[m].4 ← ACC.7~ACC.4+AC1,C=C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

DEC [m] Decrement data memory

Description: Data in the specified data memory is decremented by one

Operation: [m] ← [m]–1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

DECA [m] Decrement data memory and place result in the accumulator

Description Data in the specified data memory is decremented by one, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC ← [m]–1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

HALT Enter power down mode

Description This instruction stops program execution and turns off the system clock. The contents of the RAM and registers are retained. The WDT and prescaler are cleared. The power down bit (PD) is set and the WDT time-out bit (TO) is cleared.

Operation PC ← PC+1
PD ← 1
TO ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0	1	–	–	–	–

INC [m] Increment data memory

Description Data in the specified data memory is incremented by one

Operation [m] ← [m]+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

INCA [m] Increment data memory and place result in the accumulator

Description Data in the specified data memory is incremented by one, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC ← [m]+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

JMP addr Directly jump

Description Bits 0~10 of the program counter are replaced with the directly-specified address unconditionally, and control is passed to this destination.

Operation PC ← addr

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

MOV A,[m] Move data memory to the accumulator

Description The contents of the specified data memory are copied to the accumulator.

Operation ACC ← [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

MOV A,x Move immediate data to the accumulator

Description The 8-bit data specified by the code is loaded into the accumulator.

Operation ACC ← x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

MOV [m],A Move the accumulator to data memory

Description The contents of the accumulator are copied to the specified data memory (one of the data memory).

Operation [m] ← ACC

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

NOP No operation

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

Operation PC ← PC+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

OR A,[m] Logical OR accumulator with data memory

Description: Data in the accumulator and the specified data memory (one of the data memory) perform a bitwise logical_OR operation. The result is stored in the accumulator.

Operation: ACC ← ACC “OR” [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

OR A,x Logical OR immediate data to the accumulator

Description: Data in the accumulator and the specified data perform a bitwise logical_OR operation. The result is stored in the accumulator.

Operation: ACC ← ACC “OR” x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

ORM A,[m] Logical OR data memory with the accumulator

Description: Data in the data memory (one of the 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)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

RET Return from subroutine

Description: The program counter is restored from the stack. This is a two cycle instruction.

Operation: PC ← Stack

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RET A,x Return and place immediate data in the accumulator

Description The program counter is restored from the stack and the accumulator loaded with the specified 8-bit immediate data.

Operation PC ← Stack
ACC ← x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RETI Return from interrupt

Description The program counter is restored from the stack, and interrupts are enabled by setting the EMI bit. EMI is the enable master (global) interrupt bit (bit 0; register INTC).

Operation PC ← Stack
EMI ← 1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RL [m] Rotate data memory left

Description The contents of the specified data memory are rotated one bit left with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
[m].0 ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RLA [m] Rotate data memory left and place result in the accumulator

Description Data in the specified data memory is rotated one bit left with bit 7 rotated into bit 0, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC.(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
ACC.0 ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RLC [m] Rotate data memory left through carry

Description The contents of the specified data memory and the carry flag are rotated one bit left. Bit 7 replaces the carry bit; the original carry flag is rotated into the bit 0 position.

Operation [m].(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
[m].0 ← C
C ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

RLCA [m] Rotate left through carry and place result in the accumulator

Description Data in the specified data memory and the carry flag are rotated one bit left. Bit 7 replaces the carry bit and the original carry flag is rotated into bit 0 position. The rotated result is stored in the accumulator but the contents of the data memory remain unchanged.

Operation ACC.(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
ACC.0 ← C
C ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

RR [m] Rotate data memory right

Description The contents of the specified data memory are rotated one bit right with bit 0 rotated to bit 7.

Operation [m].i ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
[m].7 ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RRA [m] Rotate right and place result in the accumulator

Description: Data in the specified data memory is rotated one bit right with bit 0 rotated into bit 7, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation: ACC.(i) ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
ACC.7 ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RRC [m] Rotate data memory right through carry

Description: The contents of the specified data memory and the carry flag are together rotated one bit right. Bit 0 replaces the carry bit; the original carry flag is rotated into the bit 7 position.

Operation: [m].i ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
[m].7 ← C
C ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

RRCA [m] Rotate right through carry and place result in the accumulator

Description: Data of the specified data memory and the carry flag are rotated one bit right. Bit 0 replaces the carry bit and the original carry flag is rotated into the bit 7 position. The rotated result is stored in the accumulator. The contents of the data memory remain unchanged.

Operation: ACC.i ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
ACC.7 ← C
C ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

SBC A,[m] Subtract data memory and carry from the accumulator

Description: The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the accumulator.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SBCM A,[m] Subtract data memory and carry from the accumulator

Description: The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the data memory.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SDZ [m] Skip if decrement data memory is zero

Description: The contents of the specified data memory are decremented by one. If the result is zero, the next instruction is skipped. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).

Operation: Skip if ([m]–1)=0, [m] ← ([m]–1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SDZA [m] Decrement data memory and place result in ACC, skip if zero

Description: The contents of the specified data memory are decremented by one. If the result is zero, the next instruction is skipped. The result is stored in the accumulator but the data memory remains unchanged. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).

Operation: Skip if ([m]–1)=0, ACC ← ([m]–1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SET [m] Set data memory

Description Each bit of the specified data memory is set to one.

Operation $[m] \leftarrow$ FFH

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SET [m].i Set bit of data memory

Description Bit “i” of the specified data memory is set to one.

Operation $[m].i \leftarrow 1$

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SIZ [m] Skip if increment data memory is zero

Description The contents of the specified data memory are incremented by one. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).

Operation Skip if ([m]+1)=0, $[m] \leftarrow$ ([m]+1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SIZA [m] Increment data memory and place result in ACC, skip if zero

Description The contents of the specified data memory are incremented by one. If the result is zero, the next instruction is skipped and the result is stored in the accumulator. The data memory remains unchanged. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).

Operation Skip if ([m]+1)=0, ACC $\leftarrow$ ([m]+1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SNZ [m].i Skip if bit "i" of the data memory is not zero

Description: If bit "i" of the specified data memory is not zero, the next instruction is skipped. If bit "i" of the data memory is not zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).

Operation: Skip if [m].i≠0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SUB A,[m] Subtract data memory from the accumulator

Description: The specified data memory is subtracted from the contents of the accumulator, leaving the result in the accumulator.

Operation: $ACC \leftarrow ACC+\overline{[m]}+1$

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SUBM A,[m] Subtract data memory from the accumulator

Description: The specified data memory is subtracted from the contents of the accumulator, leaving the result in the data memory.

Operation: $[m] \leftarrow ACC+[\overline{m}]+1$

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SUB A,x Subtract immediate data from the accumulator

Description: The immediate data specified by the code is subtracted from the contents of the accumulator, leaving the result in the accumulator.

Operation: $ACC \leftarrow ACC+\overline{x}+1$

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SWAP [m] Swap nibbles within the data memory

Description: The low-order and high-order nibbles of the specified data memory (one of the data memories) are interchanged.

Operation: [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SWAPA [m] Swap data memory and place result in the accumulator

Description: The low-order and high-order nibbles of the specified data memory are interchanged, writing the result to 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)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SZ [m] Skip if data memory is zero

Description: If the contents of the specified data memory are zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).

Operation: Skip if [m]=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SZA [m] Move data memory to ACC, skip if zero

Description: The contents of the specified data memory are copied to the accumulator. If the contents is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).

Operation: Skip if [m]=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SZ [m].i Skip if bit “i” of the data memory is zero

Description: If bit “i” of the specified data memory is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (two cycles). Otherwise proceed with the next instruction (one cycle).

Operation: Skip if [m].i=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

TABRDC [m] Move the ROM code (current page) to TBLH and data memory

Description: The low byte of ROM code (current page) addressed by the table pointer (TBLP) is moved to the specified data memory and the high byte transferred to TBLH directly.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

TABRDL [m] Move the ROM code (last page) to TBLH and data memory

Description: The low byte of ROM code (last page) addressed by the table pointer (TBLP) is moved to the data memory and the high byte transferred to TBLH directly.

Operation: [m] ← ROM code (low byte)
TBLH ← POM code (high byte)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

XOR A,[m] Logical XOR accumulator with data memory

Description: Data in the accumulator and the indicated data memory perform a bitwise logical Exclusive_OR operation and the result is stored in the accumulator.

Operation: ACC ← ACC “XOR” [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

XORM A,[m] Logical XOR data memory with the accumulator

Description: Data in the indicated data memory and the accumulator perform a bitwise logical Exclusive_OR operation. The result is stored in the data memory. The zero flag is affected.

Operation: [m] ← ACC “XOR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

XOR A,x Logical XOR immediate data to the accumulator

Description: Data in the the accumulator and the specified data perform a bitwise logical Exclusive_OR operation. The result is stored in the accumulator. The zero flag is affected.

Operation: ACC ← ACC “XOR” x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–