NEC

NEC Electronics (Europe) GmbH

μPD 78C05

8-BIT MICROCOMPUTER

Karlstraße 123-127 · 4000 Düsseldorf 1 · W. Germany · Telephone: (0211) 36141 · Telex: 8587419 NEC D

µPD 78C05

8-BIT MICROCOMPUTER

EVAKIT FOR µPD78C06

FEBRUARY 1981

NEC ELECTRONICS (EUROPE) GMBH

uPD78C05

8-BIT MICROCOMPUTER

DESCRIPTION

The uPD78C05 is a high-performance 8-bit microcomputer fabricated with CMOS technology. The uPD78C05 contains an 8-bit ALU, a 128 x 8 RAM, two 8-bit I/O ports, a 6-bit I/O port, an 8-bit timer/event counter with 4-bit prescaler, a serial I/O port, and three (two external and one internal) source vectored interrupt structure. It also contains a 16-bit Address bus and an 8-bit data bus for external memory (program memory, data memory, or memory mapped I/O) up to 64K bytes.

The uPD78C05 has stand-by capability (STOP/HALT) for its power-down.

The uPD78C05 is applicable for hand-held computer, etc. requring low power consumption .

The uPD78C05 is compatible with the uPD78C06 (uCOM-87) and used as evaluation chip for uCOM-87LC.

FEATURES

Powerful 101 Instruction Set

Instruction Cycle Time : 4 us

Data Memory : 128W x 8

Direct Addressing Capability up to 64kB External Memory

Powerful Addressing Modes Capability

Multi-level Stack

Vectored Interrupts (External : 2, Internal : 1)

On-chip 8-bit Timer with 4-bit Prescaler

46 I/O Ports

Serial I/O Ports

Stand-by Capability (STOP/HALT mode)

Fully Bus Compatible with 8080A

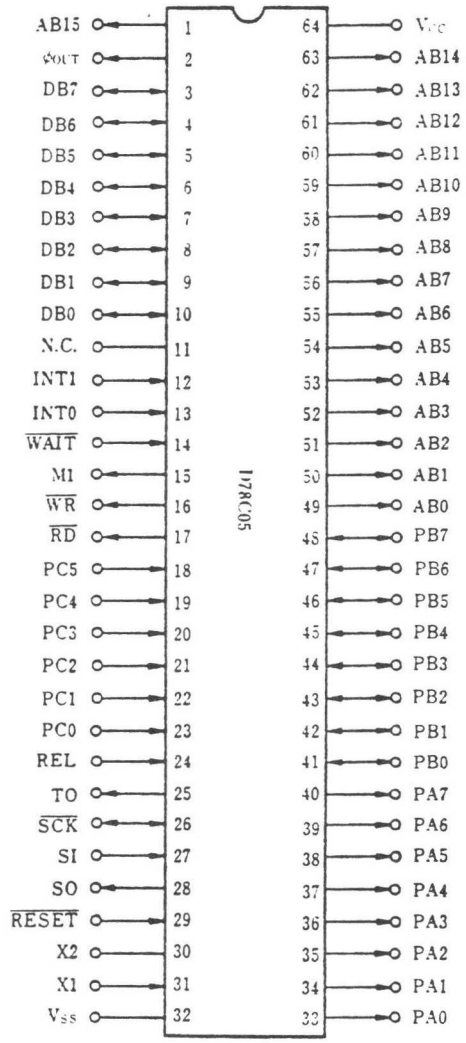
On-chip Clock Generator

Single Supply, CMOS Technology

Low Power Consumption

64 pin QUIP

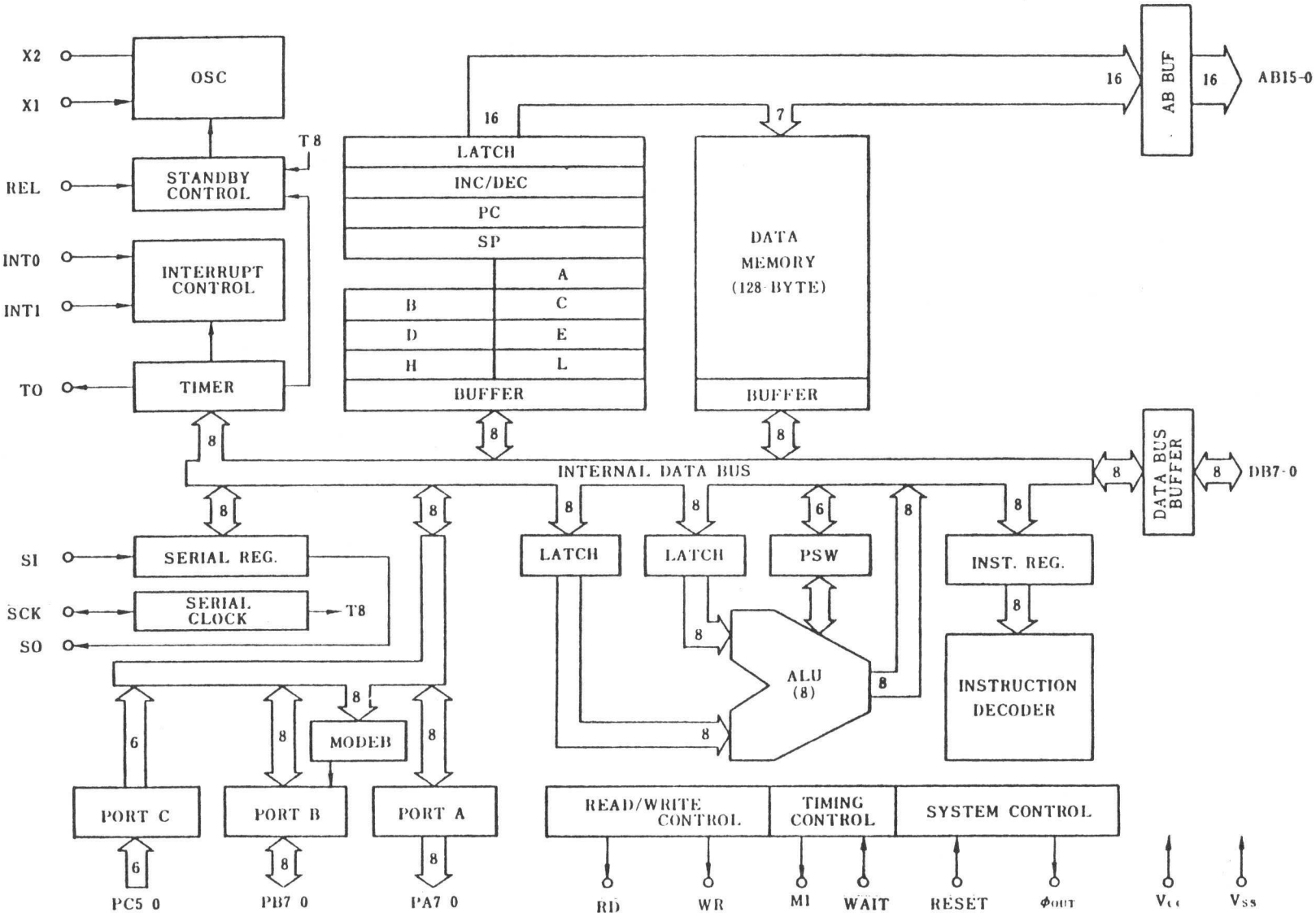
PIN CONFIGURATION



Pin Names

PA_{7-0}, PB_{7-0}, PC_{5-0}	: I/O Ports
AB_{15-0}	: Address Bus
DB_{7-0}	: Data Bus
$\overline{WAIT}$	: Wait Request
INT0, INT1	: Interrupt Request
X_2, X_1	: Xtal
$\overline{SCK}$	: Serial Clock Input/Ouput
SI	: Serial Input
SO	: Serial Output
$\overline{RESET}$	: Reset
$\overline{RD}$	: Read Strobe
$\overline{WR}$	: Write Strobe
ϕout	: Clock Output
M1	: Machine Cycle 1

uPD78C05 Block Diagram



PIN DESCRIPTION

1.1 PA_{7-0} (PortA) ••• Output

This is a 8-bit output port, and it has latch capability. With Move instructions, the data can be transferred between latch buffers and accumulator. Besides, the latched contents on the buffers can be freely handled with Logic instructions. Once the data is written on the buffers, it is kept on the buffers until another Port A handling instructions will be executed or a reset signal will be issued. Output level is TTL compatible.

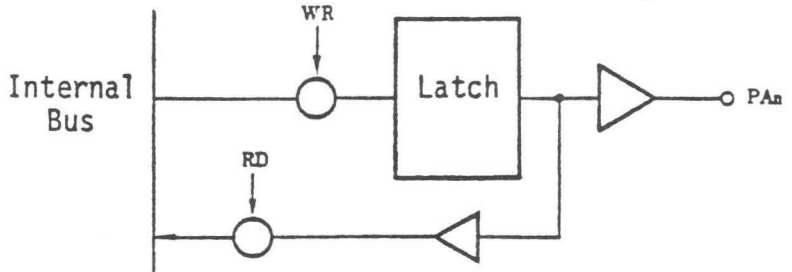


Fig. 1-1 The Port A Structure

1.2 PB_{7-0} (PortB) ••• 3-state Input/Output

This is a 8-bit input/output port, and its output has latch capability. Each line of the Port B can be independently manipulated by the MODE·B Register, and either of an input or output port can be programmed at that time. In case of being set as input port lines, or during reset, the port lines become a high impedance state. Input level is CMOS compatible and output level is TTL compatible.

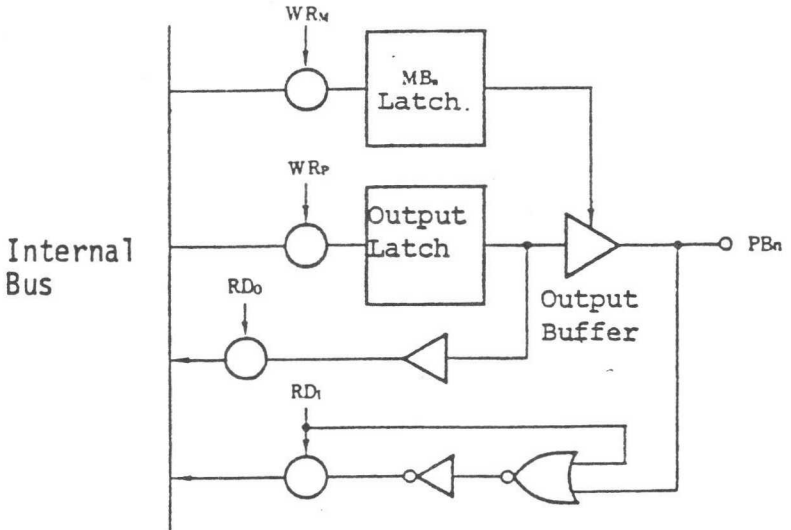


Fig. 1-2a The Port B Structure

(a) A bit of the Port B is defined as an output port (MODE·Bn=0)
In this case, an output letch becomes valid, and with move instructins. the data can be transferred between output latch buffers of the port-lines and an accumulator.

(continued)

the latched contents can be freely handled with Logic instructions. Once the data is written on the buffer, the buffer will maintains the data until another Port B handling instruction will be executed or a reset will be issued.

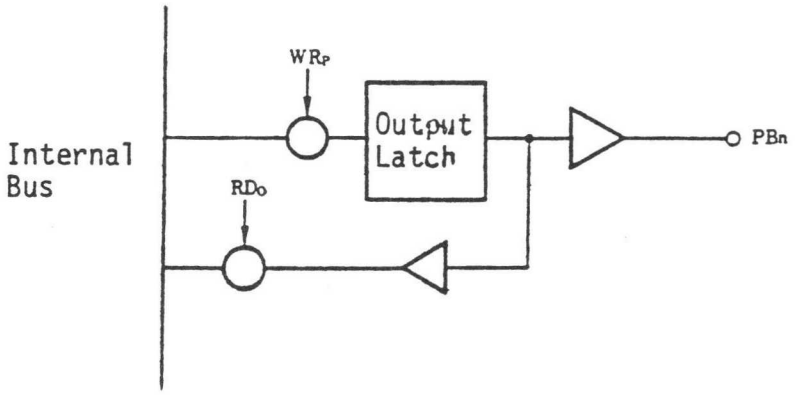


Fig. 1-2b The lines of the Port B used as output ports

(b) A bit of the Port B is defined as an input port (MODE·Bn=1)
The content of the PB lines can be loaded onto the accumulator with a move instruction. Also a write operation onto output latch buffers of the port lines are possible, too, i.e., by a move instruction, the data on the accumulator can be stored onto all the output latch buffers of the Port B, no matter which a port line is set as input or output port. However, the contents of the output latch buffers of the port lines used as input port cannot be loaded onto the accumulator, by a move instruction. Besides, since the output latch buffers of the input ports are high impedance state, the contents of the buffers do not appear on the port lines. But, if the port lines then are switched over from input mode to output mode, the contents stored in the output latch buffers mentioned in the above, can appear on the port lines, and also can be loaded onto the accumulator.

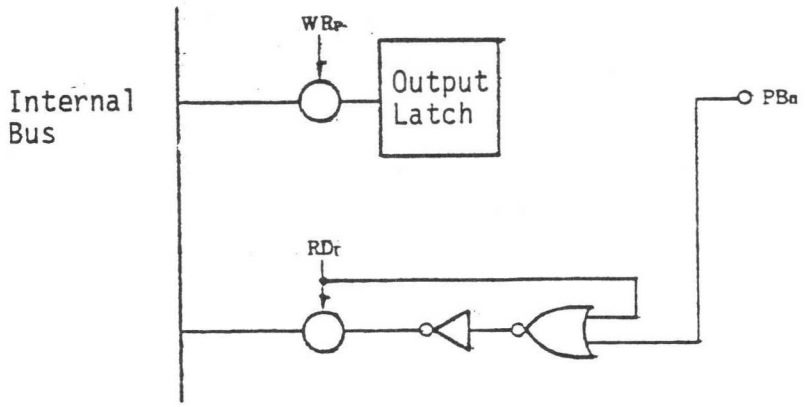


Fig. 1-2c The lines of the Port B used as input ports

(continued)

However, an actual instruction execution is done per 8-bit data. If a Port B read instruction (MOV A,PB) is executed, the contents of input lines set as input ports and the ones of output latches set as output ports are loaded onto the accumulator. If a Port B write instruction (MOV PB,A, etc.) is executed, the contents of the output latch buffers set as input ports are not replaced with another data by the instruction execution, and accordingly, the write data corresponding to these bits are disregared. Thus, the write data operation is executed only to the output latches set as output ports.

1.3 PC_{5-0} (Port C) ••• Input

This is a 6-bit input port with pull-up resistors. Input data to this port can be test by test instruction, and also moved to least significant 6-bit of accumulator. Input level is CMOS compatible. This port is fit for key-input port.

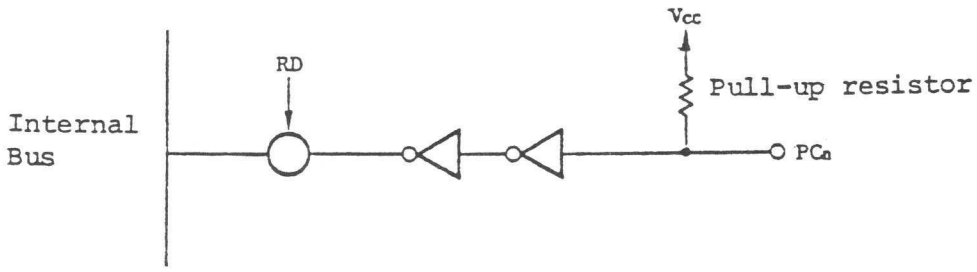


Fig. 1-3 The Port C Structure

1.4 $\overline{WR}$ (Write Strobe) ••• Output

This is used as a strobe signal for a write operation for an external memory or I/O. This is at the high impedance state in inactive condition.

1.5 $\overline{RD}$ (Read Strobe) ••• Output

This is used as a strobe signal for a read operation for an external memory or I/O. This is at the high impedance state in inactive condition.

1.6 TO (Timer Out) ••• Output

The square wave is output from this line. Its cycle time is half of a count time of the internal timer. It gows on low level after reset.

1.7 DB_{7-0} (Data Bus) ••• 3-state Input/Output

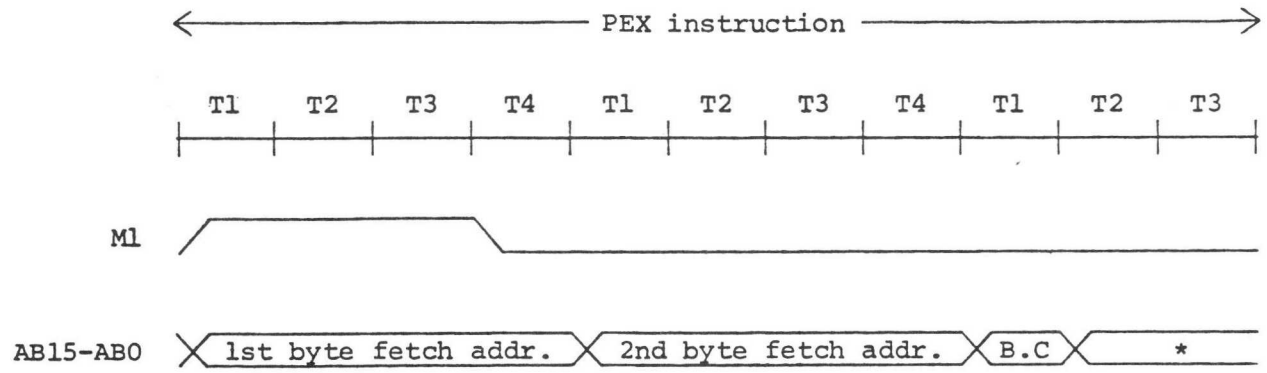
This is an 8-bit bi-directional data bus. The data move between an external memory or I/O STOP, and accumulator is done through this data bus. During an input, HALT, and RESET, the output of the data bus goes a high impedance state. Input/Output level are TTL compatible.

1.8 AB15-AB0 (Address Bus) ••• Output

These lines are 16-bit address bus and memory addresses with location 0 - 65,407, including interanl ROM addresses (0 - 4,095) in uPD78C06, are appeared on the bus.

The uPD78C05 AB lines are unlike the uPD78C06 PE lines, and have no port structure (no latching capability). When the port E output instruction (PEX) is executed at the uPD78C05, the register pair BC is output on the AB lines only during one clock cycle at ninth clock from the start of the instruction (T1-state at third machine cycle), and the memory address information are output at its before and after states.

The following diagram is the timing at PEX instruction execution.



* : 2nd byte fetch addr.

The identification code of PEX instruction and the strobe signal (STB) at the uPD78C06 are not available in the uPD78C05, so that when using the BC data for external use the contents of the data bus should be sampled and decoded to judge if the instruction code is PEX or not. If it is PEX, then count the clock (ϕout) and make the strobe signal and latch the BC data to the external register by it.

1.9 M1 (Machine Cycle 1) ••• Output

The M1 signal indicates to the external devices that the present machine cycle is the 1st one of each instruction and it is output from T1 to T3 during the 1st operation code fetch cycle.

The M1 output is used for single-step execution, break operation, and the latching control of BC data in PEX instruction execution.

1.10 $\overline{\text{WAIT}}$ (Wait Request) ••• Input

If users use rather slow speed external memories or I/O devices in the systems, they can extend a READ/WRITE timing to meet this slow device, by providing a low level signal to this line. The $\overline{\text{WAIT}}$ signal is checked at the end of T_2. If it is low, it goes to a wait state (Tw), and it stays in the state until the $\overline{\text{WAIT}}$ goes high. Pull-up register is built-in.

1.11 INT0, INT1 (Interrupt Request) ••• Input

These are Interrupt Request Input lines. INT0 is a level-sensitive, INT1 is a rising-edge sensitive, respectively. The interrupt priority among the interrupts is shown below.

INT 0 > INT T > INT 1

Here, INTT is internal interrupt.

(a) INT 0
It is a level-sensitive interrupt input line which is high level active.

(b) INT 1
It is a rising-edge sensitive interrupt line, and it becomes valid when INT 1 input goes low to high. Subsequently, if users want to perform another interrupt on this line after an interrupt on this line is accepted, they must take it into consideration that INT 1 input should be maintained at low state a little while, and then it should go high. Unless, it does not enable another interrupt.

In order to avoid a possible mis-operation due to noise signals less than 1 us, all interrupt lines are samples with internal clocks by 1 us rate, periodically. Therefore, an interrupt request signal must have more than 2 us active (high level) time.

1.12 X_1, X_2 (Xtal)

These are conncected to the crystal for the internal clock generator circuit. The X_1 is also used as the external clock input, instead of the crystal. Input level of X_1 is CMOS compatible.

1.13 Øout (Clock Output) ••• Output

The clock of system clock frequency (1/4 of crystal frequency or X_1 external clock frequency) is placed out from this line.
It is still placed out in HALT mode, but it is fixed to high in STOP mode.

1.14 $\overline{SCK}$ (Serial Clock) ••• 3-state Input/Output

This is used as control clocks for serial input/output data.
The serial clock generated in the internal circuit is placed out in internal serial clock mode, and the external clock is input to this line in external serial clock mode. At the rising edge of $\overline{SCK}$, the data on a SI line is loaded to the Serial Register (S/P), and at the falling edge of $\overline{SCK}$, the contents of the Serial Register appear on a SO line with a bit-order from MSB to LSB.

1.15 SI (Serial Input) ••• Input

This is a serial output port, and the data on the SI is loaded to the Serial Register at the rising edge of $\overline{SCK}$. The MSB is start bit.

1.16 SO (Serial Output) ••• Output

This is a serial output port, and the data on the Serial Register appears on the SO. The MSB is srart bit.

1.17 REL (Release STOP mode) ••• Input

This is an input to release the STOP mode of stand-by function. STOP mode is released by raising the REL input high, then clock generator which has been stopped will restart.
During REL input is high, the bit 3 of Stand-by Control Register (SC3) is set to one, and it is reset to zero after REL signal returns low.
Pull-down resistor is built in.

1.18 $\overline{RESET}$ (Reset) ••• Input

$\overline{RESET}$ is an input to initialize the uPD78C06. The low level signal over 8 us (at 4 MHz operation) to this line cause the system reset, and the uPD78C06 goes to the following condition;

(continued)

- All interrupt mask register bits are set, and the all interrupt sources are masked.
- An interrupt enable flag is reset, and all interrupts are disabled.
- All interrupt request flags are reset, accordingly all pending interrupts are reset, too.
- MODE·B register is set to FFH, and teh Port B lines become an input mode.
- The bit 6 of Serial Mode Register (SM6) is set, and the serial clock is in external mode.
- The bit 3 of Stand-by Control Register (SC3) is reset, other bits are set. Both the HALT and STOP mode are released.
- PSW are all reset to zero.
- PC is loaded with 0000H.
- The Prescaler and Upcounter in timer circuits are cleared.
- All TIMER·REG bits are set to FFH.
- All Timer Mode Register (TTM) bits are set. As a result, TO goes to low and the timer circuit bocomes the mode of adding PRESCALER 0.
- All Port A outputs goes to low.
- Data bus (DB7·DB0) goes to high impedance.
- So output goes to low.
- $\overline{WR}$, $\overline{RD}$ output go to high.
- Other internal registers and RAM, etc. in CPU are not specified.

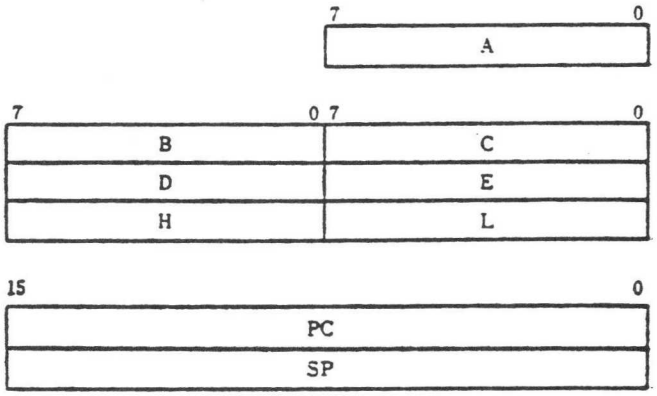
In order to avoid a possible mis-operation due to noise signals less than 4 us, this line does not accepts less than 4 us level signals as valid ones. Accordingly, $\overline{RESET}$ is accepted asynchronously and exactly by more than 8 us low level.
Following $\overline{RESET}$ goes to high, program starts from location 0000H.

2. INTERNAL BLOCK FUNCTION DESCRIPTION

2.1 Registers

This mainly consists of the seven 8-bit registers and two 16-bit registers as below.

Fig. 2-1 Register Configuration



(a) General Purpose Registers (B, C, D, E, H, L)

In addition to a role of auxiliary registers for the accumulator, these registers also have a Data Pointer capability if they are used as pair-register (BC, DE, HL). There is an auto-increment/decrement addressing mode capability for the pair-registers of DE, HL, and it can contribute to increase the program efficiency.

(c) Accumulator (A)

Since the uCOM-87LC has an accumulator based architecture, all the data operation are done through the accumulator.

(d) Program Counter (PC)

This is a 16-bit register to maintain the address information of a program step which should be executed, next. According to a number of bytes needed for an instruction which is going to be fetched, it is automatically incremented, usually. However, in case of executing a branch instruction, an immediate data or content of a register appears on the PC. If a reset signal is issued, or in stop mode the PC is reset to 0000(16).

(e) Stack Pointer (SP)

The Stack Pointer is a 16-bit register and is used to maintain a top of the address information of the stack area (Last-In-First-Out Style). The content of the SP is decremented if a CALL or PUSH instruction is executed or interrupt happens, and it is incremented if an RETURN or POP instruction is executed.

2.2 Arithmetic Logic Unit (ALU)

This is used to perform a arithmetic and logic operation such as binary addition/subtraction, decimal adjust, logic and compare operation, and rotation or digit-rotation etc.

2.3 Memory

The uPD78C05 can directly address the memory up to 64k bytes. Except on-chip ROM (0-4095), any memory location can be used as either of RAM or ROM, freely. The memory map of the uPD78C05 is shown on the next page. In the specific memory area, the Reset/Stop mode Restart Address, Interrupt Start Address, Call Table etc. are involved. External memory and on-chip RAM area can be used as data memory (RAM), program memory (ROM), and/or working registers, freely.

(a) Interrupt Start Address

Interrupt Source	Starting Address
INT0	4(0004H)
INTT	8(0008H)
INT1	16(0010H)

Since each interval among interrupt addresses is not the same, it is necessary to put an adequate interrupt service program for pre-treatment of data prior to interrupts.

(b) Call Address Table

This is used to store the call-address of each one-byte Call instruction (CALT) up to 64 call-addresses over location 128-255.

(c) External Special Use Memory Area (location 0-4095)

In location 0-255 the starting address of the Reset/Stop mode release, the interrupt starting addresses, and the call table for one-byte call instruction (CALT) are allocated, then programmer should do mapping in consideration of it.
The location 2048-4095 are directly addressed by 2-byte call instruction (CALF).
In the location 0-4095 programmer can place the memory (or memory mapped I/O), and can access it by using the address bus (AB_{15}-AB_0), data bus (DB_7-DB_0), and $\overline{RD}$ or $\overline{WR}$ signal.
In this area programmer can store both program and data.

(d) Internal Data Memory Area (65408-65535)

There is on-chip 128 bytes RAM area over location 65408-65535.

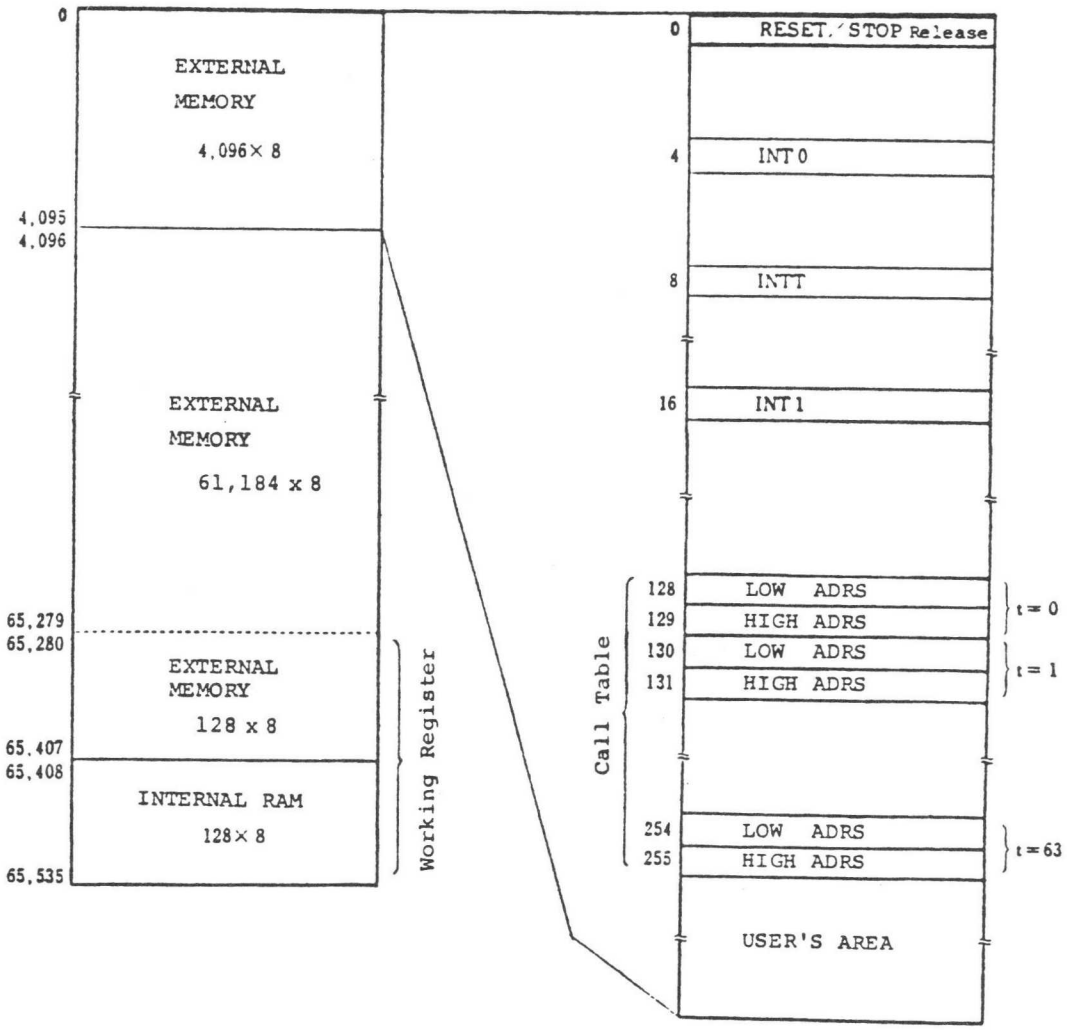
(e) External Extension Memory Area (location 4096-65407)

In this area programmer can store both program and data, and can access its data as in location 0-4095.
Note that the 128 bytes at location 65280 to 65407 can be used as the working register.

(f) Working Register Area

On the location 65280 to 65535 (internal : 128 bytes ; external : 128 bytes), the working registers which contain the 256 bytes area can be located.

Fig. 2-2 The Memory Map



2.4 Serial Interface

Serial interface section (see Fig. 2-3) consists of Serial Input (SI) line, Serial Output (SO) line, Serial Clock ($\overline{SCK}$) input/output line, an 8-bit Serial Register (S/P), an octal counter, a R-S flip-flop used for transfer control, and some gates. When the bit 6 of Serial Mode Register (SM6) is 0, $\overline{SCK}$ becomes internal clock mode fixed to a half of system clock frequency (if fosc=4MHz, then $\overline{SCK}$ is fixed to 500KHz), however, when the SM6 is 1, it becomes external clock mode, and operates with DC to 500KHz external clock. Accordingly, the transfer operation in internal clock mode performed synchronously with constant frequency, and in external clock mode it performed synchronously with variable frequency.

A transmitting data is set to serial register by MOV S, A instruction, then the octal counter is reset and serial transfer is triggered by SIO instruction. At every falling edge of $\overline{SCK}$, the contents of serial register are shift, and shift-out data are placed to SO line with starting bit of MSB.

While the $\overline{SCK}$ is low the data on SI line is loaded in continuously, and then latched to serial register at the rising edge of the $\overline{SCK}$. Lide this both the input and output of serial data are performed by same $\overline{SCK}$.

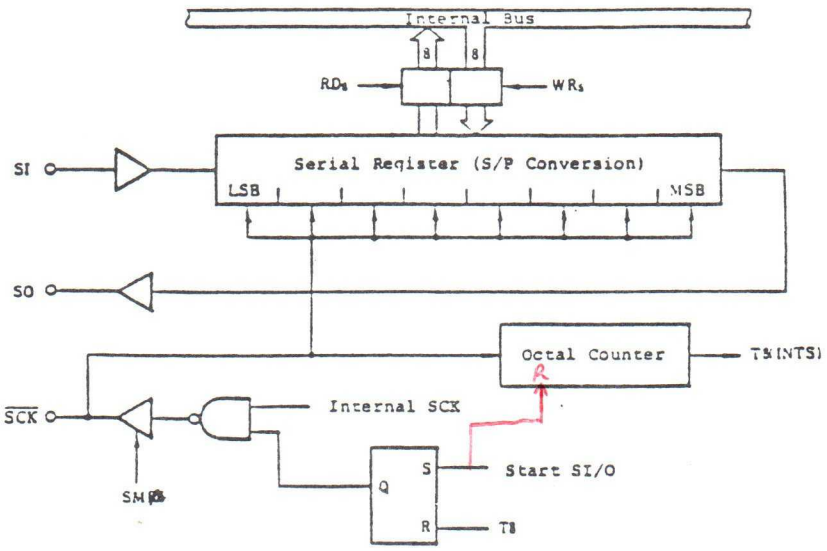


Fig. 2-3 The Block Diagram of Serial Ports

After occuring eight $\overline{SCK}$ pulses and completing 8-bit serial data trnsfer, the carry T8 is generated from the octal counter and it sats the interrupt request flag (INTFS). But uPD78C06 has no serial interrupt, then INTFSis checked by only test instruction (SKNIT FS).

Ininternal $\overline{SCK}$ mode, as T8 signal resets the control flip-flop, the following transfer after completing 8-bit trasfer are disabled until next SIO instrucdtion will be giben.

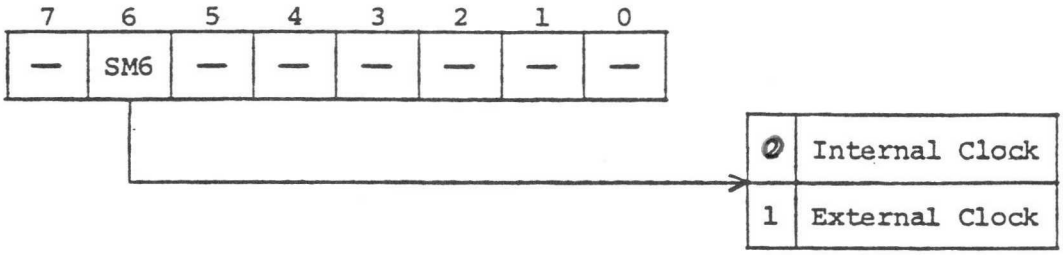
Accordingly, the data transfer should be restarted by SIO instruction with the next conditions. In case of data reception, after receiving the data from serial register by MOV A,S instruction, and in case of data transmission, after setting the data to serial register by MOV S,A instruction.

T8 is also generated in case of external $\overline{SCK}$ mode, however, it has no effect to control the external $\overline{SCK}$, so that it is necessary to control the number of $\overline{SCK}$ by the external $\overline{SCK}$ source side.
In case of external $\overline{SCK}$ mode, the trigger by SIO instruction are not required basically, but to avoid a mis-operation by noise on $\overline{SCK}$ line, the data transfer should be restarted by SIO instruction which resets the octal counter, after setting or receiving the data to/from serial register as in internal $\overline{SCK}$, after conpleting 8-bit transfer.
$\overline{RESET}$ input cause the SO to low level and the $\overline{SCK}$ is set to external clock mode.

2.5 Serial Mode Register (SM)

This is an 1-bit register used to specify the serial clock source (internal or external) as the $\overline{SCK}$, SM is set to 1 by $\overline{RESET}$ input, then the external clock is selected as serial clock source, and cleared to 0 by move instruction (MOV SM,A ; A=X0XXXXXX), then the internal clock is selected. SM (SM6) is referenced to bit 6 of accumulator. See following format.

Fig. 2-4 Serial Mode Register Format



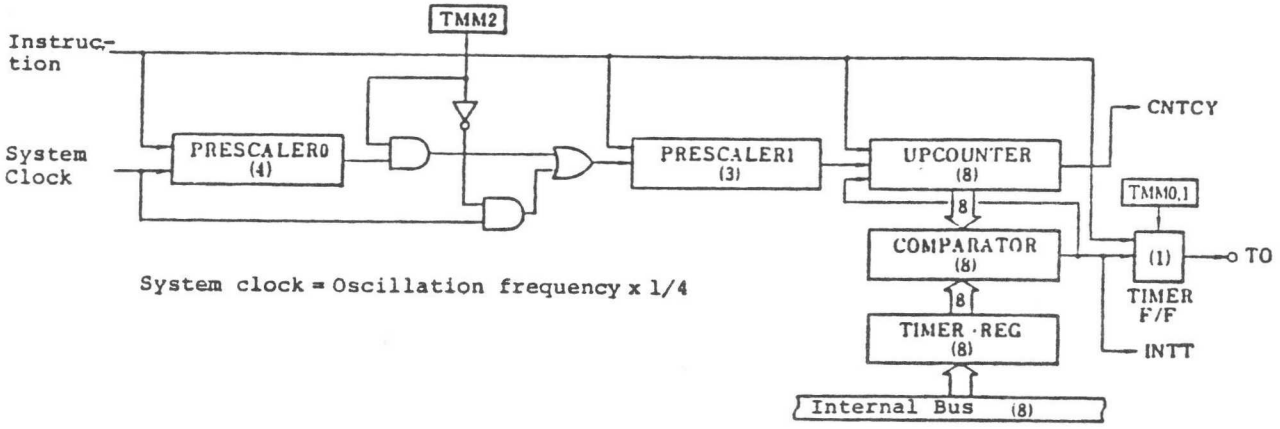
2.6 Timer

This is a programmable 8-bit interval timer with prescaler. It consists of TIMER·REG (8-bit), PRESCALER0 (4-bit), PRSCALER1 (3-bit), UPCOUNTER (8-bit), COMPARATOR (8-bit), and TIMER F/F.
Count time and TO output are controlled by Timer Mode Register (TMM). It can count 8 usec to 2 msec with resolution of 8 usec (TMM2=0),or 128 usec to 32 msec with resolut;ion of 128 usec (TMM2=1).
At first set the count value to the TIMER REG by MOV TM, A instruction, then initialize the PRESCALER0, 1, TIMER F/F, UPCOUNTER and start timer.
UPCOUNTER is incremented at every 8us (TMM2=0) or 128us (TMM2=1).
COMPARATOR always comparates the contents of UPCOUNTER with TIMER·REG, and it generates match signal (internal interrupt ; INTT) when they are matched. The match signal clears the content of UPCOUNTER, and restart the countup. Accordingly, this timer operates as the interval timer which generates repetitive interrupt with the interval of count time specified by count value of TIMER·REG. When a timer interrupt is generated in HALT mode, the HALT mode is released.

(continued)

Note that the timer interrupt is disabled by setting the bit 1 of interrupt mask register (MK1) to 1. The content of TIMER F/F which is changed its state by every match signal from COMPARATOR are placed on the TO line, so that the TO signal becomes the square wave with its half cycle time equivalent to count time.
This output is suitable for driving a piezo buzzer, etc.

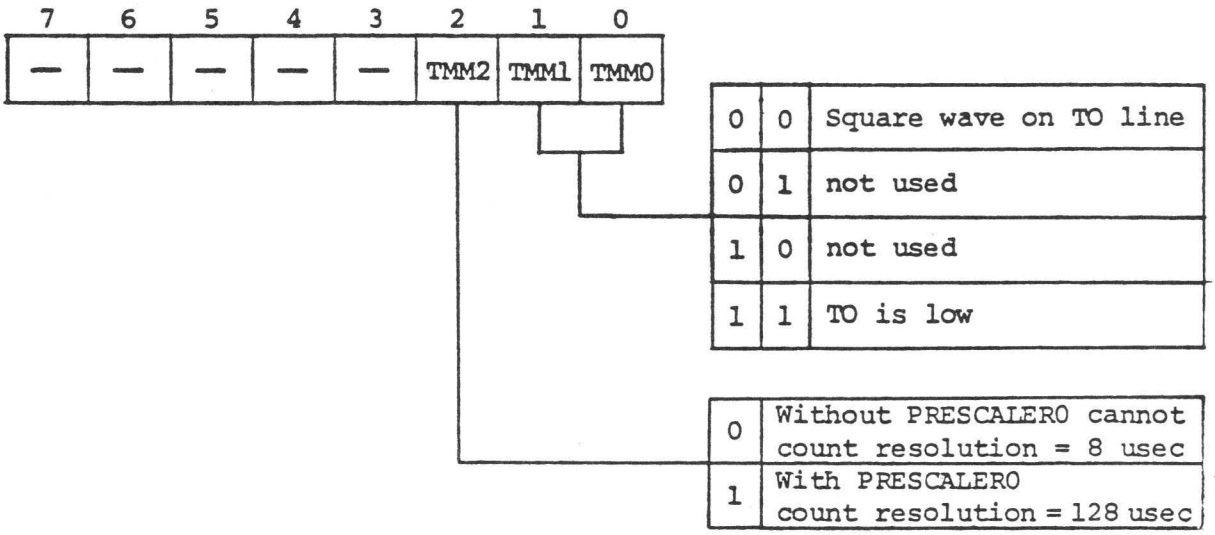
Fig. 2-5 Timer



2.7 Timer Mode Register (TMM)

This is a 3-bit register used to control the timer operation.
Timer is transferred to/from the accumulator by move instruction.
It is referrenced to bit 0-2 of accumulator.
Bit 0 and 1 of timer mode register (TMM0,1) control to enable or disable the square wave output of TO, and bit 2 (TMM2) controls to add the PRESCALER0 or not.

Fig. 2-6 Timer Mode Register Format



2.8 MODE•B

This is an 8-bit Register used for programming the input/output modes of the Port B. With Move instructions (MOV MB, A), programmer can determine the contents of the Port B, freely. They can program each bit line as either of input or output mode, individually.
If one of the bit of MODE B register is set to one, then corresponding bit of PB line becomes input mode, and if it is reset to zero, then corresponding bit of PB line becomes output mode. All MODE•B register bits are set to one by $\overline{\text{RESET}}$ input.

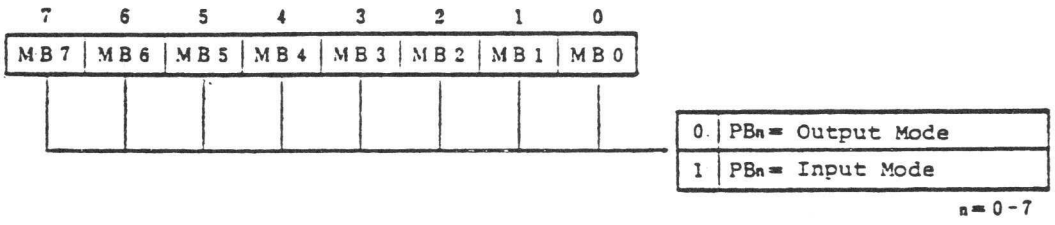


Fig. 2-7 Mode•B Register Format

2.9 Program Status Word (PSW)

It contains the six flags which are set or reset by the results of instruction execution. Two of these flags (Z, CY) can be tested with instructions. The contents of the PSW are automatically saved onto the Stack at interrupt occurrence (External Interrupt, Internal Interrupt), and are retrieved by RETI instruction. All the contents are reset to 0 by the $\overline{\text{RESET}}$ input or STOP mode.

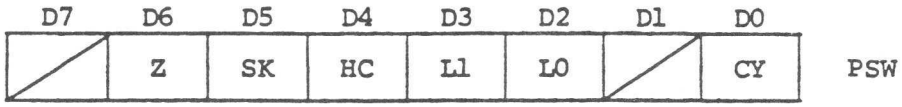


Fig. 2-8 PSW Format

(a) Z (Zero)

If an result of execution is 0, it is set to 1, otherwise it is reset to 0. By instruction, it can be tested.

(b) SK (Skip)

If the Skip condition is just complete, it is set to 1, otherwise it is reset to 0.

(c) HC (Half Carry)

If a Carry from the Bit 3 of the accumulator occurs as a result of operation, it is set to 1, otherwise it is reset to 0.

(d) L1

If there is a string of MVI A, byte instructions in a program, it is set to 1, otherwise it is reset to 0.

(e) L0

If there is a string of MVI L, byte; LXI H, word instructions in a program, it is set to 1, otherwise it is reset to 0.

(f) CY (Carry)

If a Carry from the Bit 7 of the ALU occurs as a operation result, it is set to 1, otherwise it is reset to 0. By instructions, it can be tested.

(g) The Correlation of flags with instruction executions

By execution of 18 kinds of ALU instruction, rotation instructions and Carry manipulation instructions, the flags of the uCOM-87 LC are affected as shown in the following table.

Table 2-1 Flag Operation

Operation					D6	D5	D4	D3	D2	D0
reg, memory		immediate		skip	Z	SK	HC	L1	L0	CY
ADD ADC SUB SBB	ADDX ADCX SUBX SBBX	ADI ACI SUI SBI			↕	0	↕	0	0	↕
ANA ORA XRA	ANAX ORAX XRAX	ANI ORI XRI	ANIW ORIW		↕	0	●	0	0	●
ADDNC SUBNB GTA LTA	ADDNCX SUBNBX GTAX LTAX	ADINC SUINB GTI LTI	 GTIW LTIW		↕	↕	↕	0	0	↕
	ONAX OFFAX	ONI OFFI	ONIW OFFIW		↕	↕	●	0	0	●
NEA EQA	NEAX EQAX	NEI EQI	NEIW EQIW		↕	↕	↕	0	0	↕
INR DCR	INRW DCRW				↕	↕	↕	0	0	●
DAA					↕	0	↕	0	0	↕
RLL, RLR					●	0	●	0	0	↕
RLD, RRD					●	0	●	0	0	●
STC					●	0	●	0	0	1
CLC					●	0	●	0	0	0
		MVI A, byte			●	0	●	1	0	●
		MVI L, byte LXI H, word			●	0	●	0	1	●
				SKN SKNIT	●	↕	●	0	0	●
				RETS	●	1	●	0	0	●
All other instructions					●	0	●	0	0	●

↕ ..... Flag Affected (Set or Reset)
1 ..... Flag Set
0 ..... Flag Reset
● ..... Flag Not Affected

2.10 Stand-by Control Register (SC)

This is a 5-bit register used to control the stand-by function, i.e. STOP or HALT mode. Bit 0-2,4 of Stand-by Control Register (SC0-2,4) can be set or reset by loading it with the contents of accumulator with MOV SC, A instruction. Bit 1 and 3 (SC1,3) can be mored to the corresponding bits a ccumulator with MOV A, SC instruction, and other bits of accumulator (A0, A2, A4-7) will be undefined.
RESET resets SC3, and sets other bits (SC0-2,4).

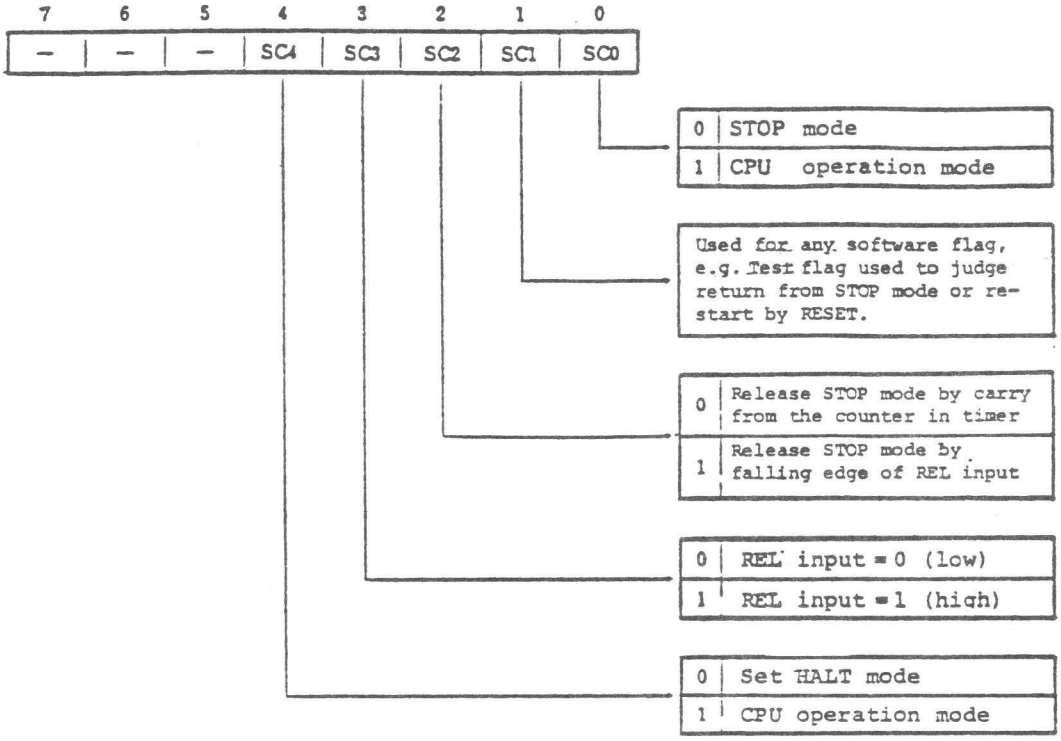


Fig. 2-9 Stand-by Control Register Format

Note that SC3 is not a control bit, but status bit to reflect a state of REL input directly. Accordingly, it will be possible to check the off-chattering of STOP mode release key signals, etc. by checking the content of SC3.

2.11 Interrupt Control Block

There are two external interrupts and one internal interrupt shown at the following, and all these are vectored interrupts.

	Interrupt Source	Starting Address	Priority
External	INT 0 (Level	4	1
	INT 1 (Rising Edge)	16	3
Internal	INTT (Match on timer comparator)	8	2

The interrupt Control Block contains INTERRUPT REQUEST Register, MASK Register, PRIORITY CONTROL, TEST CONTROL, INTERRUPT ENABLE F/F etc.

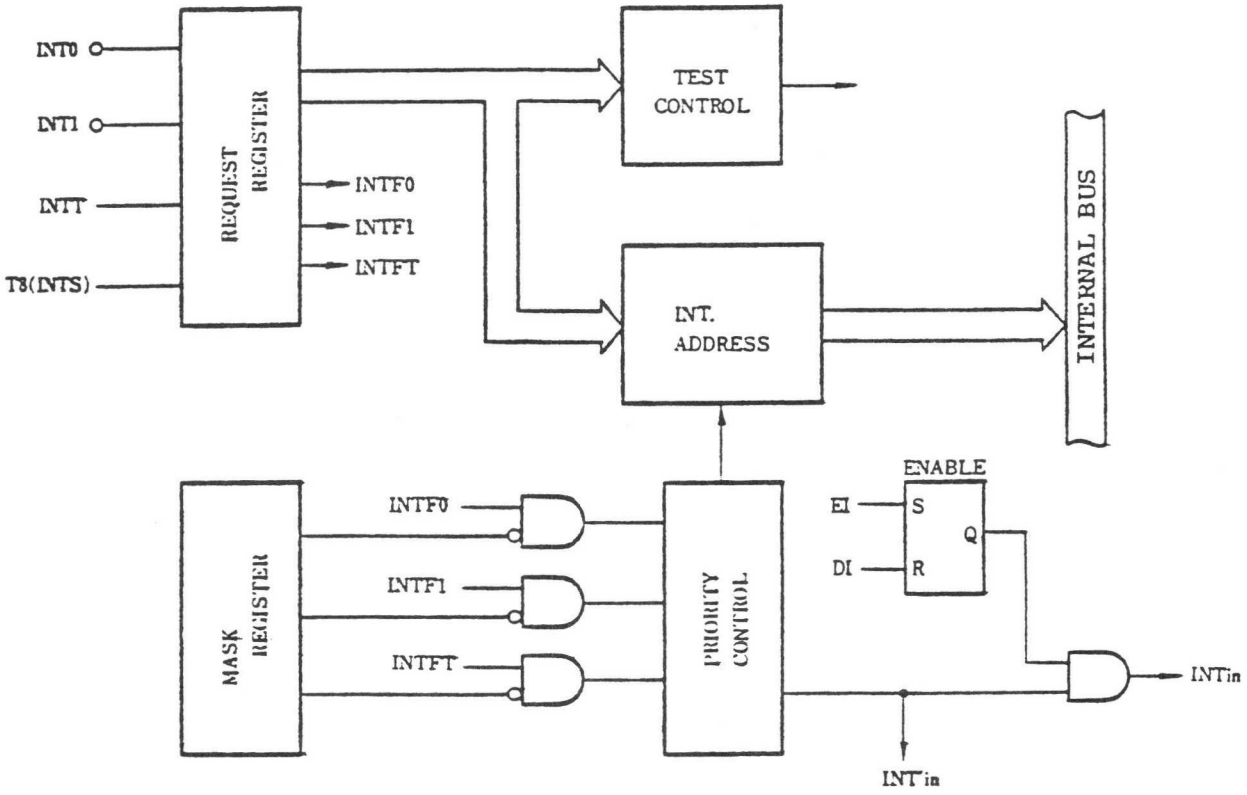


Fig. 2-10 Interrupt Control Block

(a) INTERRUPT REQUEST Register

It contains 3 kinds of interrupt request flags which are set to 1 by each interrupt, individually. By a system reset or STOP mode, all the flags are reset.

• INTF0

This is a flag set by an external level interrupt (INT0). If the line receives a high level signal, this flag is set to 1. If it receives a low level signal, the flag is reset.

• INTF1

By the rising edge of an input signal to the INT1, the flag is set to 1.

• INTFT

This flag is set to 1 by match signal from the comparator in Timer.

• INTFS

If a Serial Register completes the receipt of 8-bit data through a SI line or if it completes the transmission of the data through a SO line, the flag is set to 1.
However no interrupt is driven by this, and INTFS is checked by on SKNIT instruction.

(b) MASK Register

It contains 3 bits mask flags corresponding to each interrupt. By instructions, each flag can be set or reset, freely. If a MASK bit is 1, the corresponding bit is masked.

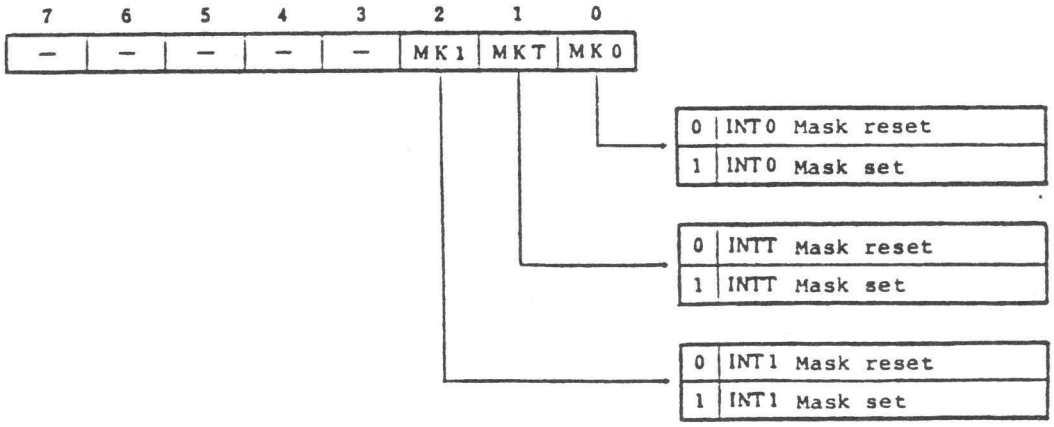


Fig. 2-11 Mask Register Format

(c) PRIORITY CONTROL Circuit

This is the circuit used for controlling the interrupt priority among the interrupts mentioned in the above. If more than two interrupts occur in the system at once, the uCOM-87LC accepts an interrupt which has a higher (or highest) priority than others.

INT 0 > INTT > INT 1

(d) TEST CONTROL Circuit

This circuit operates by when instructions of checking a status of interrupt request flags (INTF0, 1, T) or test flag showing serial transfer completion (INTFS) are executed.

(e) INTERRUPT ENABLE F/F

It is set by EI instruction, and is reset by DI instruction. Once any one of interrupts is accepted, it is reset. If this F/F is set to 1, it means 'Interrupt Enable'. If it is reset to 0, it means 'Interrupt Disable'. By a RESET input or STOP mode, it is also reset.

3. Interrupt Procedure

Each interrupt request is processed by the following procedure.

1) Interrupt request flags are checked at T1 timing of first machine cycle of every instruction, and when it is set interrupt sequence will start. However the masked interrupt requests are not checked.

2) If more than two interrupts are set at same time, then their priorities will be checked and the highest priority interrupt request is acknowledged, and others are pended.

(INT0 > INTT > INT1)

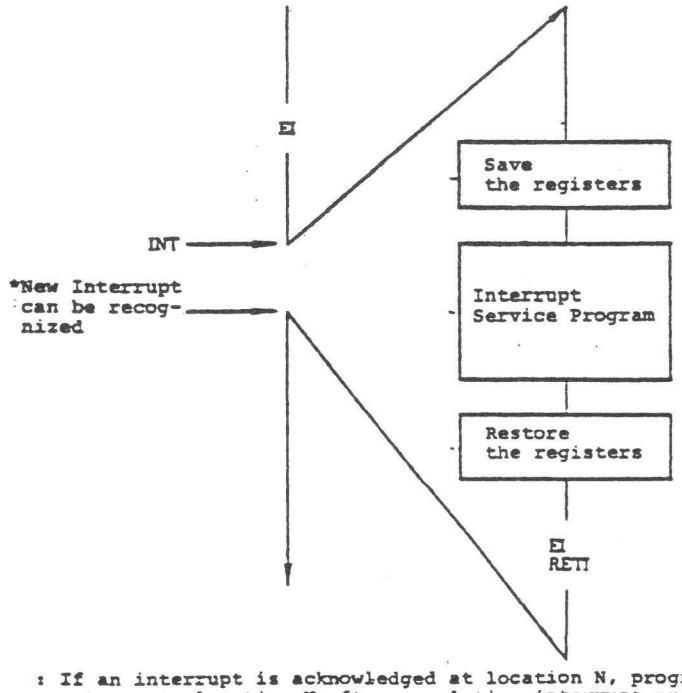
3) Reset interrupt enable flip-flop, then all interrupts are disabled.

4) Reset the interrupt request flag which is acknowledged, except for level-activated interrupt (INT0).

5) Save the PSW, upper byte of PC, lower byte of PC onto the stack in this sequence.

6) Jump to each interrupt starting address.

After execution of interrupt service program, it should be performed to return to the address where the interrupt was acknowledged. First, registers or flags except for PSW are restored and interrupt enable flag is set by EI instruction. Then the lower byte of PC, the upper byte of PC, and PSW are restored in this sequence by RETI instruction. To avoid the stack overflow following interrupt will be recognized after two instruction are executed subsequent to EI instruction. This means a new interrupt can be recognized after completing an execution of RETI instruction following EI instruction and as a result completing to restore from stack.



* If an interrupt is acknowledged at location N, program returns to location N after completing interrupt procedure.

Fig. 3-1 Interrupt Sequence

4. Stand-by operation

Stand-by function is used to lower the power consumption in stnad-by condition, and there are two types of it, HALT mode and STOP mode. Stand-by control block diagram are shown in Figure 4-1. It is specified to HALT mode by to reset the bit 4 of Stand-by Control Register(SC4) to 0, or STOP mode by to reset the bit 0 (SC0) to 0. HALT mode can be released by one of the external interrupt (INT0, 1), timer interrupt (INTT), carry from the serial clock counter (T8), and RESET signal. When HALT mode is released by an interrupt, program jumps to corresponding interrupt starting address in EI condition, or steps to the instruction following HALT mode setting instruction (MOV SC,A) in DI condition.
Yet in HALT mode the masking function is active, so that programmer can choose an interrupt source for release use.
When HALT mode is released by T8, it returns the program control to an instruction of the main routine which is placed immediately after the instruction activating the HALT mode.
When HALT mode is released by RESET, normal reset operation will be performed and program jumps to location 0.
STOP mode can be released by REL or RESET signal, and there are two types of releasing way (shown following) by REL signal as the content of bit 2 of Stnad-by Control Register (SC2).

1) In case of SC2 = 0

   Start the oscillator and timer by rising edge of REL signal, and start to provide the internal check after occurring four carry (i.e. after 1,024 count) from UPCOUNTER in timer, then program will start at location 0.

2) In case of SC2 = 1

   Start the oscillator by rising edge of REL signal and inhibit to provide the internal clock during REL signal raised to high, and restart to provide the internal clock after REL signal returns to low, then program will start at location 0.

   When STOP mode is released by RESET, normal reset operation will be performed, so that the oscillator will start at the falling edge of RESET, and program will start at location 0 by next rising edge. Accordingly, RESET should be held at low level sufficient time for oscillator to become stable.

Table 4-1 HALT Mode and STOP Mode

Parameter	HALT mode	STOP mode
Oscillator	Run	Stop
Internal System Clock	Stop	
Timer	Run	
TIMER · REG	Hold	Set
UPCOUNTER, PRECALER 0, 1	Run	Cleared
Serial Interface		Run *1
Serial Clock	Hold	Hold
Interrupt Control Circuit	Run	Stop
Interrupt Enable Flag	Hold	Reset
INT0, INT1 Input	Active	Inactive
INTT		—
T8 (INTFS)		—
MASK Register	Hold	Set
Pending Interrupts (INTFX)		Reset
REL Input	Inactive	Active
$\overline{\text{RESET}}$ Input	Active	
On-Chip RAM	Hold	Hold
Output Latch in Port A, B, E		
Program Counter (PC)	Hold	Cleared
Stack Pointer (SP)		Hold
General Registers (A, B, C, D, E, H, L)		
Program Status Word (PSW)		Reset
MODE B·Register		Hold
Stand-by Control Register (SC0-SC3)		
Stand-by Control Register (SC4)		Set
Timer Mode Register (TMM0,1)		Hold
Timer Mode Register (TMM2)		Set
Serial Mode Register (SM)		Hold
Data Bus (DB0-7)	High-Z	High-Z
$\overline{\text{RD}}$, $\overline{\text{WR}}$ Output	High	High

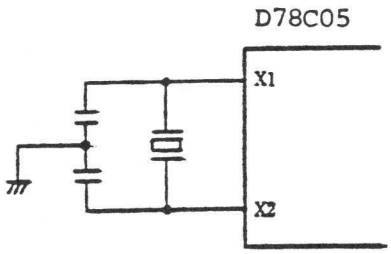
Note 1. Serial clock counter are running and T8 is generated, however, there are no effect by it.

5. Clock Driver Circuit

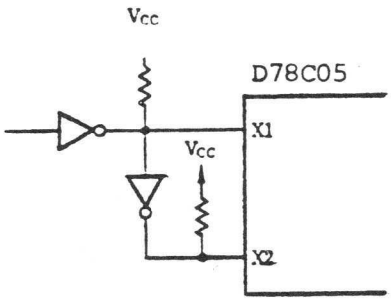
You may drive the clock input (X1, X2) of the uPD78C05 with a crystal, or an external clock source. The driving frequency must be four times the desired internal system clock frequency.

Fig. 5-1 Clock Driver Circuit

a) Crystal



b) External Source



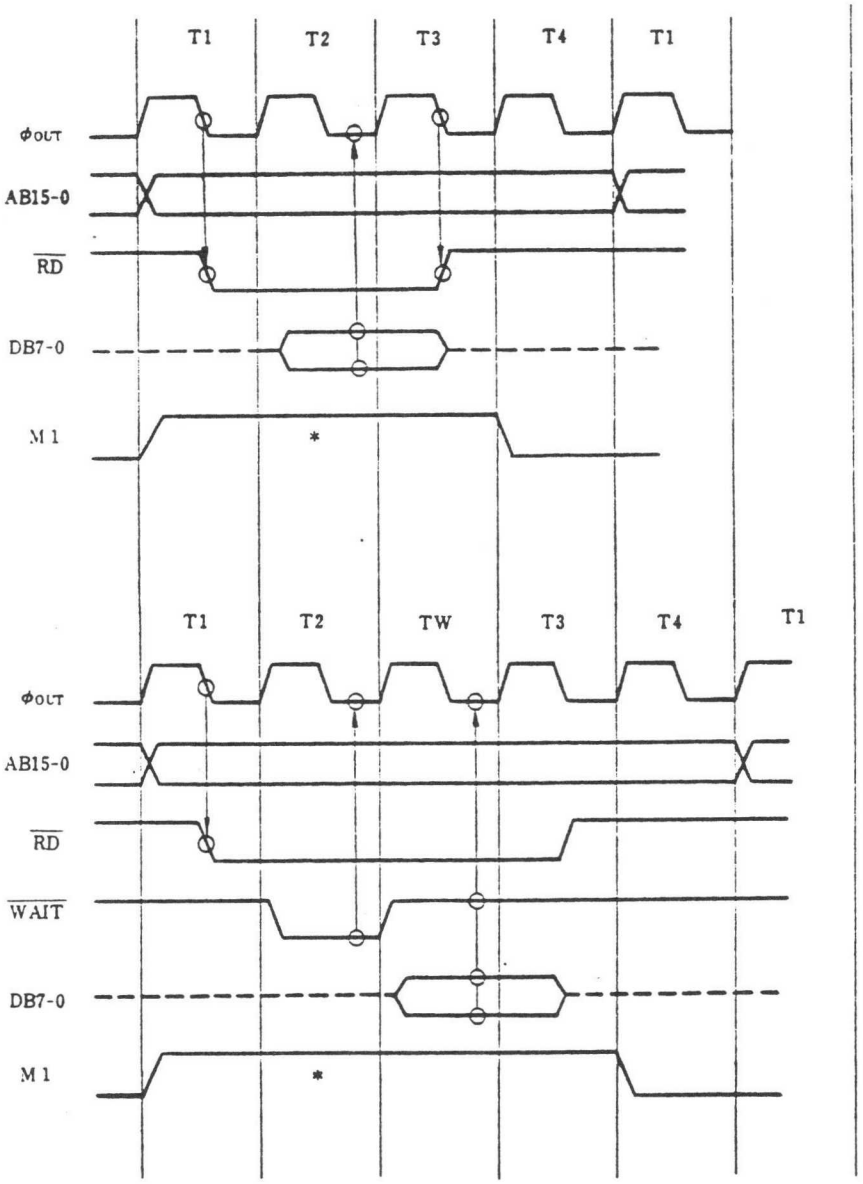
6. Bus Interface Timing

Input/Output timing of the Address Bus (AB_{15-0}), Data Bus (DB_{7-0}), $\overline{RD}$, $\overline{WR}$, $\overline{WAIT}$, M1 are shown in Fig. 6-1 to 6.3.

6.1 ·OP Code fetch

Fig. 6-1 OP code fetch (1st or 2nd) timing

(1) No wait



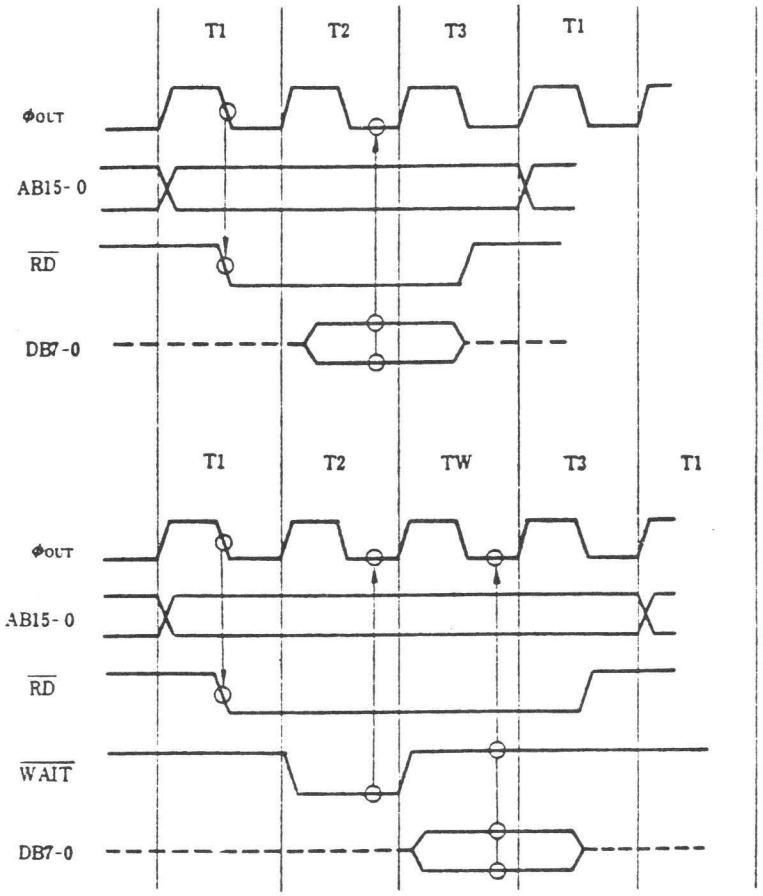
(2) 1 wait

* : M1 is not output at 2nd OP code fetch.

6.2 Memory read

(1) No wait

Fig. 6-2 Memory read timing

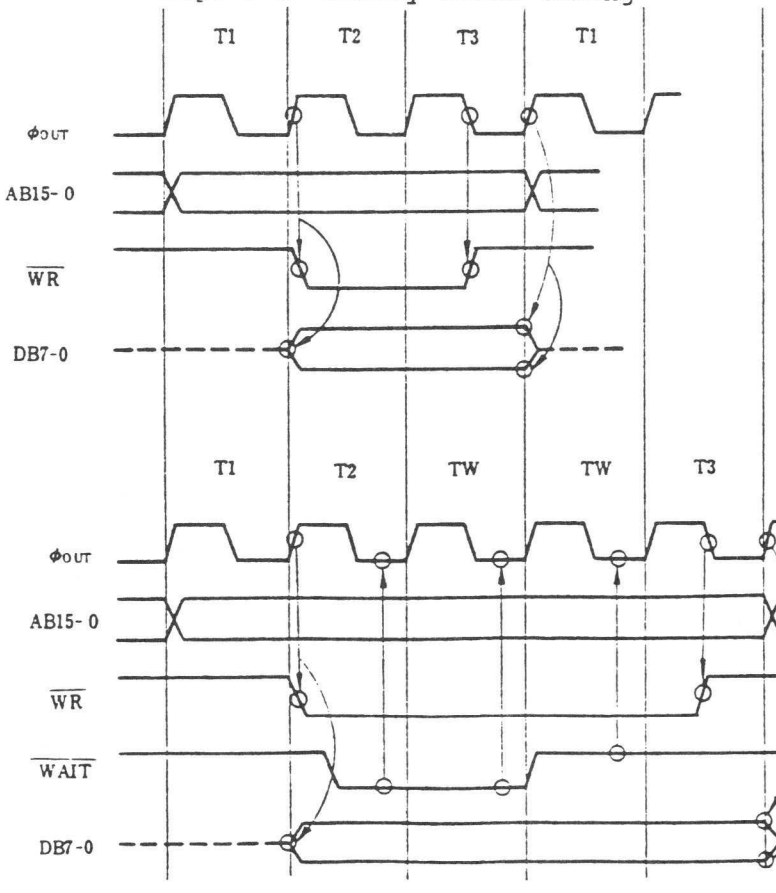


(2) 1 wait

6.3 Memory write

(1) No wait

Fig. 6-3 Memory write timing



(2) 1 wait

7. String of Peculiar Three Instrucitions

When more than one certain instruction among the three occurs in sequence, only the first instruction encountered will be executed. The remainder of the instructions in teh string will be ignored and just be replaced with idle clocks number of same clocks required for usual execution of these instructions. There are two groups of the instructions which have the above peculiar feature, and these two groups are independent of, each other.

```
Group A : MVI  A, byte                    (L1 Flag)
Group B : MVI  L, byte ; LXI  H, word     (L0 Flag)
```

If there is a string of the instructions (MVI A, byte(belonged to the group A, the L1 flag is set to 1. If there is a string of the instructions (MVI L, byte and/or LSI H, word) belonged to the group B, the L0 flag is set to 1. An interrupt is not disabled during the execution of a string of these instructions. Since the L flags are saved to the Stack during an interrupt operation, the uCOM-87LC can judge whether or not the next instruction is a part in the string with these flags after returning from the interrupt procedure.

```
Start ——→ MVI  A, 0       ;A←0
          MVI  A, 1       ;NOP(7 CLOCKS), L1=1
          MVI  A, 2       ;NOP(7 CLOCKS), L1=1
          MVI  L, 0AH     ;L←0AH
          MVI  L, 0BH     ;NOP(7 CLOCKS), L0=1
Interrupt ——→
          MVI  L, 0CH     ;NOP(7 CLOCKS), L0=1
          LXI  H, 00H     ;NOP(10 CLOCKS), L0=1
```

8. uPD78C05 Instruction Set

The instruction set of the uPD78C05 is compatible with uPD78C06 (uCOM-87LC) except for clock cycle.

8.1 Symbols/Description on Operand

Symbols	Descriptions
r	A, B, C, D, E, H, L
r1	B, C, D, E, H, L
r2	A, B, C
sr	PA PB MK MB TM S TMM SM SC
sr1	PA PB PC MK S TMM SC
sr2	PA PB PC MK
rp	SP, B, D, H
rp1	V, B, D, H
rpa	B, D, H, D+, H+, D−, H−
wa	8 bit immediate data
word	16bit "
byte	8 bit "
bit	3 bit "
if	F0, F1, FT, FS,
f	CY, Z

(Notes)

1. At sr~sr2, the symbols of 'PA', 'PB', etc. stand for the following, respectively:

   PA = PORTA, PB = PORTB, PC = PORTC, MK = MASK•reg, MB = MODE•B, TM = TIMER•REG, S = SERIAL I/O, TMM = TIMER MODE REG, SM = SERIAL MODE REG, SC = STANDBY CONTROL REG

2. At rp~rp1, the 'SP', 'B', etc. stand for the following, respectively:

   SP = STACK POINTER, B = BC, D = DE, H = HL, V = FFH•A

3. At rpa, the 'B', 'D', etc. stand for the following, respectively:

   B = (BC), D = (DE), H = (HL), D+ = $(DE)^+$, H+ = $(HL)^+$, D− = $(DE)^-$, H− = $(HL)^-$

4. At if, the 'F0', 'F1', etc. stand for the following, respectively:

   F0 = INTF0, F1 = INTF1, FT = INTFT, FS = INTFS

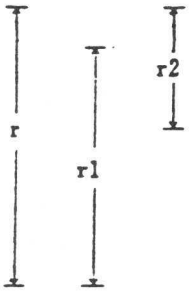
5. At f, the 'CY', 'Z', stand for the following, respectively.

   CY = CARRY, Z = ZERO

8.2 The description of the symbols on Operation Codes is as follows:

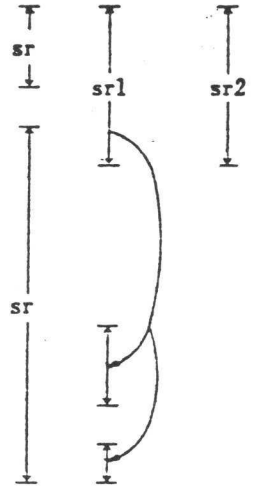
r

R_2	R_1	R_0	reg
0	0	0	
0	0	1	A
0	1	0	B
0	1	1	C
1	0	0	D
1	0	1	E
1	1	0	H
1	1	1	L



sr

S_3	S_2	S_1	S_0	special-reg
0	0	0	0	PORT A
0	0	0	1	PORT B
0	0	1	0	PORT C
0	0	1	1	MASK
0	1	0	0	MODE · B
0	1	0	1	—
0	1	1	0	TIMER · REG
0	1	1	1	—
1	0	0	0	SERIAL · I/O
1	0	0	1	TIMER MODE REG.
1	0	1	0	SERIAL MODE REG.
1	0	1	1	STANDBY CONTROL REG.

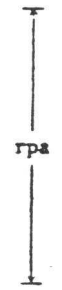


rp

P_1	P_0	reg·pair
0	0	SP
0	1	BC
1	0	DE
1	1	HL

rpa

A_2	A_1	A_0	addressing
0	0	0	—
0	0	1	(BC)
0	1	0	(DE)
0	1	1	(HL)
1	0	0	$(DE)^+$
1	0	1	$(HL)^+$
1	1	0	$(DE)^-$
1	1	1	$(HL)^-$



rp1

Q_1	Q_0	reg·pair
0	0	FFH.A
0	1	BC
1	0	DE
1	1	HL

if

I_2	I_1	I_0	INTF
0	0	0	INTF 0
0	0	1	INTFT
0	1	0	INTF 1
0	1	1	—
1	0	0	INTFS

f

F_2	F_1	F_0	flag
0	1	0	CY
1	0	0	Z

	Mnemonics	Operand	Op Codes B1	Op Codes B2	Op Codes B3	Op Codes B4	Clock Cycle	Functions	Skip Condition
8-Bit Data Move Instructions	MOV	r1, A	0 0 0 1 1 $R_2R_1R_0$				4	r1←A	
	MOV	A, r1	0 0 0 0 1 $R_2R_1R_0$				4	A←r1	
	MOV	sr, A	0 1 0 0 1 1 0 1	1 1 0 0 $S_3S_2S_1S_0$			10	sr←A	
	MOV	A, sr1	0 1 0 0 1 1 0 0	1 1 0 0 $S_3S_2S_1S_0$			10	A←sr1	
	MOV	r, word	0 1 1 1 0 0 0 0	0 1 1 0 1 $R_2R_1R_0$	Low Adrs	High Adrs	17	r←(word)	
	MOV	word, r	0 1 1 1 0 0 0 0	0 1 1 1 1 $R_2R_1R_0$	Low Adrs	High Adrs	17	(word)←r	
	MVI	r, byte	0 1 1 0 1 $R_2R_1R_0$	← Data →			7	r←byte	
	STAW	wa	0 0 1 1 1 0 0 0	← Offset →			10	(FFH. wa)←A	
	LDAW	wa	0 0 1 0 1 0 0 0	← Offset →			10	A←(FFH. wa)	
	STAX	rpa	0 0 1 1 1 $A_2A_1A_0$				7	(rpa)←A	
	LDAX	rpa	0 0 1 0 1 $A_2A_1A_0$				7	A←(rpa)	
	SBCD	word	0 1 1 1 0 0 0 0	0 0 0 1 1 1 1 0	Low Adrs	High Adrs	20	(word)←C, (word+1)←B	
	SDED	word		0 0 1 0 1 1 1 0			20	(word)←E, (word+1)←D	
	SHLD	word		0 0 1 1 1 1 1 0			20	(word)←L, (word+1)←H	
	SSPD	word		0 0 0 0 1 1 1 0			20	(word)←SP_L, (word+1)←SP_H	

	Mnemonics	Operand	Op Codes				Clock Cycle	Functions	Skip Condition
			B 1	B 2	B 3	B 4			
16-Bit Data Move Instructions	LBCD	word	0 1 1 1 0 0 0 0	0 0 0 1 1 1 1 1	Low Adrs	High Adrs	20	C←(word), B←(word+1)	
	LDED	word		0 0 1 0 1 1 1 1			20	E←(word), D←(word+1)	
	LHLD	word		0 0 1 1 1 1 1 1			20	L←(word), H←(word+1)	
	LSPD	word		0 0 0 0 1 1 1 1			20	SP_L←(word), SP_H←(word+1)	
	PUSH	rp1	0 1 0 0 1 0 0 0	0 0 Q_1Q_0 1 1 1 0			17	(SP−1)←$rp1_H$, (SP−2)←$rp1_L$	
	POP	rp1	0 1 0 0 1 0 0 0	0 0 Q_1Q_0 1 1 1 1			14	$rp1_L$←(SP), $rp1_H$←(SP+1) SP←SP+2	
	LXI	rp, word	0 0 P_1P_0 0 1 0 0	←——Low Byte ——	High Byte		10	rp←word	
Arithmetic Instructions	ADD	A, r	0 1 1 0 0 0 0 0	1 1 0 0 0 $R_2R_1R_0$			8	A←A+r	
	ADDX	rpa	0 1 1 1	1 1 0 0 0 $A_2A_1A_0$			11	A←A+(rpa)	
	ADC	A, r	0 1 1 0	1 1 0 1 0 $R_2R_1R_0$			8	A←A+r+CY	
	ADCX	rpa	0 1 1 1	1 1 0 1 0 $A_2A_1A_0$			11	A←A+(rpa)+CY	
	SUB	A, r	0 1 1 0	1 1 1 0 0 $R_2R_1R_0$			8	A←A−r	
	SUBX	rpa	0 1 1 1	1 1 1 0 0 $A_2A_1A_0$			11	A←A−(rpa)	
	SBB	A, r	0 1 1 0	1 1 1 1 0 $R_2R_1R_0$			8	A←A−r−CY	
	SBBX	rpa	0 1 1 1	1 1 1 1 0 $A_2A_1A_0$			11	A←A−(rpa)−CY	
	ADDNC	A, r	0 1 1 0	1 0 1 0 0 $R_2R_1R_0$			8	A←A+r	No Carry
	ADDNCX	rpa	0 1 1 1	1 0 1 0 0 $A_2A_1A_0$			11	A←A+(rpa)	No Carry
	SUBNB	A, r	0 1 1 0	1 0 1 1 0 $R_2R_1R_0$			8	A←A−r	No Borrow
	SUBNBX	rpa	0 1 1 1	1 0 1 1 0 $A_2A_1A_0$			11	A←A−(rpa)	No Borrow

	Mnemonics	Operand	Op Codes B1	B2	B3	B4	Clock Cycle	Functions	Skip Condition
Logic Instructions	ANA	A, r	01100000	10001 $R_2R_1R_0$			8	A←A∧r	
	ANAX	rpa	0111	10001 $A_2A_1A_0$			11	A←A∧(rpa)	
	ORA	A, r	0110	10011 $R_2R_1R_0$			8	A←A∨r	
	ORAX	rpa	0111	10011 $A_2A_1A_0$			11	A←A∨(rpa)	
	XRA	A, r	0110	10010 $R_2R_1R_0$			8	A←A∀r	
	XRAX	rpa	0111	10010 $A_2A_1A_0$			11	A←A∀(rpa)	
	GTA	A, r	0110	10101 $R_2R_1R_0$			8	A−r−1	No Borrow
	GTAX	rpa	0111	10101 $A_2A_1A_0$			11	A−(rpa)−1	No Borrow
	LTA	A, r	0110	10111 $R_2R_1R_0$			8	A−r	Borrow
	LTAX	rpa	0111	10111 $A_2A_1A_0$			11	A−(rpa)	Borrow
	ONAX	rpa	0111	11001 $A_2A_1A_0$			11	A∧(rpa)	No Zero
	OFFAX	rpa	0111	11011 $A_2A_1A_0$			11	A∧(rpa)	Zero
	NEA	A, r	0110	11101 $R_2R_1R_0$			8	A−r	No Zero
	NEAX	rpa	0111	11101 $A_2A_1A_0$			11	A−(rpa)	No Zero
	EQA	A, r	0110	11111 $R_2R_1R_0$			8	A−r	Zero
	EQAX	rpa	0111	11111 $A_2A_1A_0$			11	A−(rpa)	Zero

	Mnemonics	Operand	Op Codes B1	Op Codes B2	Op Codes B3	Op Codes B4	Clock Cycle	Functions	Skip Condition
Immediate Operation Instructions (Accumulator)	XRI	A, byte	0 0 0 1 0 1 1 0	——Data——→			7	A←A∨byte	
	ADINC	A, byte	0 0 1 0				7	A←A+byte	No Carry
	SUINB	A, byte	0 0 1 1				7	A←A−byte	No Borrow
	ADI	A, byte	0 1 0 0				7	A←A+byte	
	ACI	A, byte	0 1 0 1				7	A←A+byte+CY	
	SUI	A, byte	0 1 1 0				7	A←A−byte	
	SBI	A, byte	0 1 1 1				7	A←A−byte−CY	
	ANI	A, byte	0 0 0 0 0 1 1 1				7	A←A∧byte	
	ORI	A, byte	0 0 0 1				7	A←A∨byte	
	GTI	A, byte	0 0 1 0				7	A−byte−1	No Borrow
	LTI	A, byte	0 0 1 1				7	A−byte	Borrow
	ONI	A, byte	0 1 0 0				7	A∧byte	No Zero
	OFFI	A, byte	0 1 0 1				7	A∧byte	Zero
	NEI	A, byte	0 1 1 0				7	A−byte	No Zero
	EQI	A, byte	0 1 1 1				7	A−byte	Zero
(Special Register)	ANI	sr2, byte	0 1 1 0 0 1 0 0	1 0 0 0 1 0 S_1 S_0	Data		17	sr2←sr2∧byte	
	ORI	sr2, byte		1 0 0 1			17	sr2←sr2∨byte	
	ONI	sr2, byte		1 1 0 0			14	sr2∧byte	No Zero
	OFFI	sr2, byte		1 1 0 1			14	sr2∧byte	Zero

	Mnemonics	Operand	Op Codes B1	Op Codes B2	Op Codes B3	Op Codes B4	Clock Cycle	Functions	Skip Condition
Working Register Operation Instructions	ANIW	wa, byte	00000101	← Offset →	Data		16	(FFH.wa)←(FFH.wa)∧byte	
	ORIW	wa, byte	0001				16	(FFH.wa)←(FFH.wa)∨byte	
	GTIW	wa, byte	0010				13	(FFH.wa)−byte−1	No Borrow
	LTIW	wa, byte	0011				13	(FFH.wa)−byte	Borrow
	ONIW	wa, byte	0100				13	(FFH.wa)∧byte	No Zero
	OFFIW	wa, byte	0101				13	(FFH.wa)∧byte	Zero
	NEIW	wa, byte	0110				13	(FFH.wa)−byte	No Zero
	EQIW	wa, byte	0111				13	(FFH.wa)−by'	Zero
Increment/Decrement Instructions	INR	r2	010000R1R0				4	r2←r2+1	Carry
	INRW	wa	00100000	← Offset →			13	(FFH.wa)←(FFH.wa)+1	Carry
	DCR	r2	010100R1R0				4	r2←r2−1	Borrow
	DCRW	wa	00110000	← Offset →			13	(FFH.wa)←(FFH.wa)−1	Borrow
	INX	rp	00P1P00010				7	rp←rp+1	
	DCX	rp	00P1P00010				7	rp←rp−1	
Other Operation Instructions	DAA		01100001				4	Decimal Adjust Accumulator	
	STC		01001000	00101011			8	CY←1	
	CLC		01001000	00101010			8	CY←0	

	Mnemonics	Operand	Op Codes B1	Op Codes B2	Op Codes B3	Op Codes B4	Clock Cycle	Functions	Skip Condition
Rotation Instructions	RLD		01001000	00111000			17	Rotate Left Digit	
	RRD			1001			17	Rotate Right Digit	
	RLL	A *1		0000			8	$Am+1 \leftarrow Am$, $A_0 \leftarrow CY$, $CY \leftarrow A_7$	
	RLR	A *2		0001			8	$Am-1 \leftarrow Am$, $A_7 \leftarrow CY$, $CY \leftarrow A_0$	
Jump Instructions	JMP	word	01010100	← Low Adrs →	High Adrs		10	PC←word	
	JB		01110011				4	$PC_H \leftarrow B$, $PC_L \leftarrow C$	
	JR	word	11←jdisp1 →				10	PC←PC+1+jdisp1	
	JRE	word	0100111←	← jdisp →			13	PC←PC+2+jdisp	
Call Instructions	CALL	word	01000100	← Low Adrs →	High Adrs		16	$(SP-1) \leftarrow (PC+3)_H$, $(SP-2) \leftarrow (PC+3)_L$, PC←word	
	CALF	word	01111←	← fa →			13	$(SP-1) \leftarrow (PS+2)_H$, $(SP-2) \leftarrow (PC+2)_L$ PC15~11←00001, PC10~0←fa	
	CALT	word	10← ta →				19	$(SP-1) \leftarrow (PC+1)_H$, $(SP-2) \rightarrow PC+1)_L$ $PC_L \leftarrow (128+2ta)$, $PC_H \leftarrow (129+2ta)$	
Return Instructions	RET		00001000				10	$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$ SP←SP+2	
	RETS		00011000				10+n	$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$, SP←SP+2 PC←PC+n	Unconditional skip
	RETI		01100010				13	$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$ PSW←(SP+2), SP←SP+3	

*1 : RLL A can be alternated with RAL (no operands)

*2 : RLR A can be alternated with RAR (no operands)

	Mnemonics	Operand	Op Codes				Clock Cycle	Functions	Skip Condition
			B 1	B 2	B 3	B 4			
Skip Instruction	SKN	f *3	0 1 0 0 1 0 0 0	0 0 0 1 1 $F_2F_1F_0$			12	Skip if No Flag	f = 0
	SKNIT	if		0 0 0 1 0 I_2 I_1 I_0			12	Skip if No INT× Reset INT× if INT× = 1	if = 0
CPU Control Instructions	NOP		0 0 0 0 0 0 0 0				6	No Operation	
	EI		0 1 0 0 1 0 0 0	0 0 1 0 0 0 0 0			12	Enable Interrupt	
	DI		0 1 0 0 1 0 0 0	0 0 1 0 0 1 0 0			12	Disable Interrupt	
Register Control Instructions	SIO		0 0 0 0 1 0 0 1				6	Start (Trigger) Serial I/O	
	STM		0 0 0 1 1 0 0 1				6	Start Timer	
Input/Output Instructions	PEX		0 1 0 0 1 0 0 0	0 0 1 0 1 1 0 1			15	$PE_{15-8} \leftarrow B$, $PE_{7-0} \leftarrow C$	
	PER		0 1 0 0 1 0 0 0	0 0 1 1 1 1 0 0			12	PortE AB Mode	

*3 : SKN CY and SKN Z can be alternated with SKNC (no operands) and SKNZ (no operands) respectively.

9. Differences between the uPD78C05 and the uPD78C06

Parameters		uPD78C05	uPD78C06
4k bytes on-chip ROM		No	Yes
Internal WAIT for on-chip ROM		No	2 WAIT
Port E (Address Bus)	after reset	Address bus mode	Port mode
	latch capability	No	Yes
	PEX instruction	At only M3T1 timing AB15-8←B, AB7-0←C is executed, and at other timing the contents of the internal address bus are output.	At M3T1 timing Pe15-8←B, PE7-0←C is executed and the BC data are latched. Until the next PEX or PER instruction will be executed, the output data will not be changed.
$\overline{RD}$, $\overline{WR}$ Signal		Output for the location 0-65,407 (0000H-FF7FH).	Output for the location 4,096-65,407 (1000H-FF7FH).
M1 output		Yes	No
Pin configuration		Different	
Package		64-pin QUIP	64-pin flat

NEC

NEC Electronics (Europe) GmbH

NEC Electronics France, (Paris-Office), Tour Chenonceaux, 204, Rond Point du Pont de Sèvres, F-92516 Boulogne Billancourt,
Tel. (01) 620-6400, Telex 203 544
NEC Electronics Italiana S.r.l., Via Cardona 3, 20124 Milano, Tel. (02) 63 26 46, Telex 315 355
NEC Electronics UK, 116, Stevenston Street, New Stevenston, Motherwell ML 14 LT, United Kingdom, Tel. (06 98) 73 22 21, Telex 777 565
Hannover Office, Gutenbergstr. 4, 3012 Langenhagen, W.-Germany, Tel. (0511) 73 80 01, Telex 9 230109
Munich Office, Bayerstr. 21, 8000 München 2, W.-Germany, Tel. (0 89) 5913 64-68, Telex 5 22 971
Stuttgart Office, Solitudestr. 218, 7000 Stuttgart 31, W.-Germany, Tel. (07 11) 881119, Telex 7 252 220
Eindhoven Office, 33 Alard du Hamelstraat, 5622 CC Eindhoven, Netherlands, Tel. (0 40) 44 58 55, Telex 51 923
Stockholm Office, Box 4039, S-18304 Täby, Tel. 08-7567245, Telex 13839