TMPZ84C00AP-6 / TMPZ84C00AM-6 / TMPZ84C00AT-6
TMPZ84C00AP-8 / TMPZ84C00AM-8 / TMPZ84C00AT-8
TLCS-Z80 MPU : 8-BIT MICROPROCESSOR

1. OUTLINE AND FEATURES

The TMPZ84C00A is an 8-bit microprocessor (hereinafter referred to as MPU), which provides low power operation and high performance.

Built into the TMPZ84C00A are bus control, memory control and timing control circuits in addition to paired 6 general purpose registers, accumulator, flag registers and an arithmetic-and-logic unit.

The TMPZ84C00A is fabricated using Toshiba's CMOS Silicon gate Technology.

The pricipal functions and features of the TMPZ84C00A are as follows.

Table 1.1 Operating Frequency and Supply Current

Produce Name	Operating Frequency	Supply Current (TYP.)	
		AT RUN	AT STAND BY
TMPZ84C00AP-6/AM-6/AT-6	6MHz	15mA	0.5μA
TMPZ84C00AP-8/AM-8/AT-8	8MHz	20mA	0.5μA

060489

(1) Commands compatible with the Zilog Z80 MPU.

(2) Low power consumption

(3) DC to 8MHz operation (at 5V ± 10%)

(4) Single 5V power supply (5V ± 10%)

(5) Operating temperature (− 40°C to 85°C)

(6) Powerful set of 158 instrucitons available

(7) Powerful interrupt function

(a) Non-maskable interrupt terminal ($\overline{NMI}$)

(b) Maskable interrupt terminal ($\overline{INT}$)

The following 3 modes are selectable:

- 8080 compatible interrupt mode (interrupt by Non-Z80 family peripheral LSI) (Mode 0)
- Restart interrupt (Mode 1)

- Daisy chain structure interrupt using Z80 family peripheral LSI (Mode 2)

(8) An auxiliary resister provided to each of general purpose registers

(9) Two index registers

(10) 10 addressing modes

(11) Built-in refresh circuit for dynamic memory

(12) Molded in 40-pin DIP package (P), 40-pin SOP package (M) and 44-pin PLCC package (T).

Further, in the following text and explanations for charts and tables, hexadecimal numbers are directly used without giving an identification to explanation of address, etc. to the extent not to cause confusions.

Note : Z80 is a trademark of Zilog Inc., U.S.A.

2. PIN ASSIGNMENT AND FUNCTIONS

The pin assignment and I/O pin names and brief functions of TMPZ84C00A are shown below.

2.1 PIN ASSIGNMENT (TOP View)

The pin assignment of the TMPZ84C00A are as shown in Figure 2.1 and Figure 2.2.

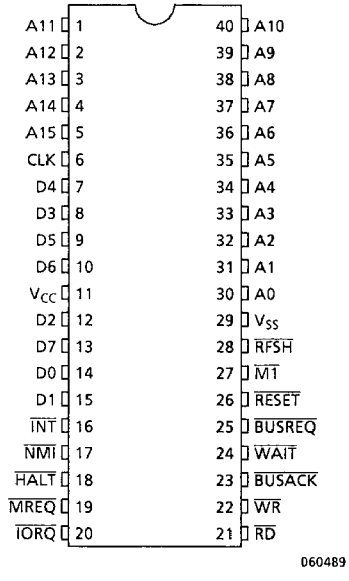


Figure 2.1 DIP, SOP Pin Assignment

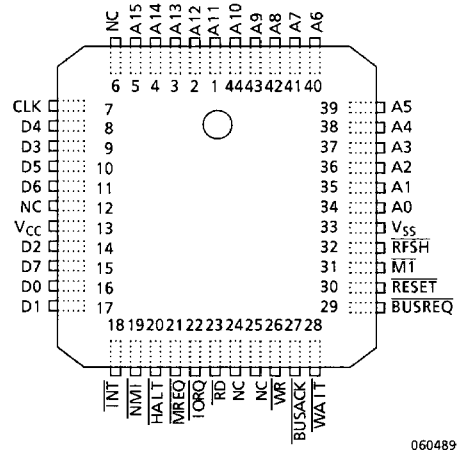


Figure 2.2 PLCC Package Pin Assignment

2.2 PIN NAMES AND FUNCTIONS

I/O pin names and functions are as shown Table 2.1.

Table 2.1 Pin names and Functions (1/2)

Pin	Q'ty (Number)	Type	Function
D0-D7	8	Input/output 3-state	The 8-bit bi-directional data bus.
A0-A15	16	Output 3-state	The 16-bit address bus. These pins specify memory and I/O port addresses. During a refresh cycle, the refresh address is output.
$\overline{M1}$	1	Output	The Machine Cycle 1 signal. In an operation code fetch cycle, this pin goes "0" with the $\overline{MREQ}$ signal. At the execution of a 2-byte operation code, this pin goes "0" for each operation code fetch. In a maskable interrupt acknowledge cycle, this pin goes "0" with the $\overline{IORQ}$ signal.
$\overline{RD}$	1	Output 3-state	The Read signal. It indicates that the MPU is ready for accepting data from memory or I/O device. The data from the addressed memory or I/O devices is gated by this signal onto the MPU data bus.
$\overline{WR}$	1	Output 3-state	The Write signal. This signal is output when the data to be stored in the addressed memory or I/O device is on the data bus.
$\overline{MREQ}$	1	Output 3-state	The Memory Request signal. When the execution address for memory access is on the address bus, this pin goes "0". During a memory refresh cycle, this pin also goes "0" with $\overline{RFSH}$ signal.
$\overline{IORQ}$	1	Output 3-state	The Input/Output Request signal. This pin goes "0" when the address for an I/O read or write operation is on the low-order 8 bits (A0 through A7) of the address bus. The $\overline{IORQ}$ signal is also output with the $\overline{M1}$ signal at interrupt acknowledge to tell an I/O device that the interrupt response vector can be placed on the data bus.
CLK	1	Input	The Single-phase Clock Input. When the clock input is placed in the DC state (continued "1" or "0" level), this pin stops operating and holds the state of that time.

060489

(2/2)

Pin	Q'ty (Number)	Type	Function
$\overline{RESET}$	1	Input	The Reset signal input. $\overline{RESET}$ signal is used for initialization MPU and must be kept in active state ("0") for a period of at least 3 clocks.
$\overline{INT}$	1	Input	The Maskable Interrupt signal. An interrupt is caused by the peripheral LSI. An interrupt is acknowledged when the interrupt enable flip-flop (IFF) is set to "1" by software. The $\overline{INT}$ pin is normally wire-ORed and requires an external pullup resistor for these applications.
$\overline{WAIT}$	1	Input	The Wait Request signal. This signal indicates to the MPU that the addressed memory or I/O device is not ready for data transfer. As long as this signal is "0", the MPU is in the Wait state.
$\overline{BUSREQ}$	1	Input	The bus Request signal. The $\overline{BUSREQ}$ signal forces the MPU address bus, data bus, and control signals $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ to be placed in the high-impedance state. This signal is normally Wire-ORed and requires an external pullup resistor for these applications.
$\overline{BUSACK}$	1	Output	The Bus Acknowledge signal. In response to the $\overline{BUSREQ}$ signal, the $\overline{BUSACK}$ signal indicates to the requesting peripheral LSI that the MPU address bus, data bus, and control signals $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and WR have been put in the high-impedance state.
$\overline{HALT}$	1	Output	The Halt signal. This pin goes "0" when the MPU has executed a Halt instruction and is in the Halt state.
$\overline{RFSH}$	1	Output	The refresh signal. When the dynamic memory refresh address is on the low-order 8 bits of the address bus, this signal goes "0" . At the same time, the $\overline{MREQ}$ signal also goes active ("0").
$\overline{NMI}$	1	Input	The Non-maskable Interrupt Request signal. This interrupt request has a higher priority than the maskable interrupt and is not dependent on the interrupt enable flip-flop (IFF) state.
NC (PLCC only)	4	–	Not connected internaly. Please use by open.
V_{CC}	1	power supply	+ 5 V
V_{SS}	1	power supply	0 V

060489

3. FUNCTIONAL DESCRIPTION

The system configuration, functions and basic operation of the TMPZ84C00A are described here.

3.1 BLOCK DIAGRAM

The block diagram of the internal configuration is shown in Figure 3.1.

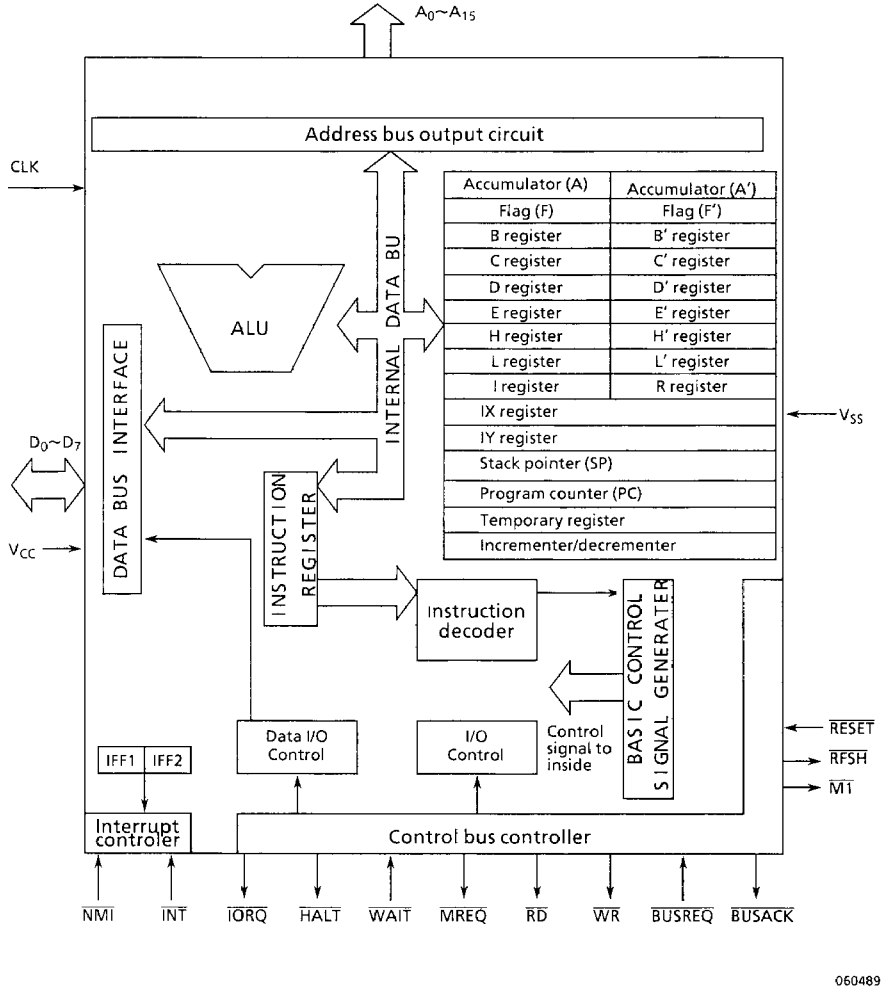


Figure 3.1 Block Diagram

3.2 SYSTEM CONFIGURATION

The MPU has the configuration shown in Figure 3.1. The address signal is put on the address bus via the address buffer. The data bus is controlled for input or output by the data bus interface. Both the address and data buses are put in the high-impedance state by the $\overline{BUSEQ}$ signal input to make them available for other peripheral LSIs. The Opcode read from memory via the data bus is written to the instruction register. This Opcode is decoded by the instruction decoder. According to the result of the decoding, control signals are sent to the relevant devices. Receiving these control signals, the ALU performs arithmetic operations. The register array temporarily hold the information required to perform operation.

The following describes the MPU's main components and functions which the user must understand to operate the TMPZ84C00A.

[1] Internal Register Groups

The configuration of the internal register groups is as follows:

(1) Main registers

A, F, B, C, D, E, H, L

(2) Alternate registers

A', F', B', C', D', E', H', L'

(3) Special purpose registers

I, R, IX, IY, SP, PC

Figure 3.3 shows the configuration of the internal register groups. The register groups, each being of a static RAM, consists of eighteen 8-bit registers and four 16-bit registers. The following describes the function of each register:

(1) Main registers (A, F, B, C, D, E, H, L)

(a) Accumulator (A)

The accumulator is an 8-bit register used for arithmetic and data transfer operations.

(b) Flag register (F) (see Fig. 3.2)

The flag register is an 8-bit register to hold the result of each arithmetic operation. Actually, the 6 of the 8 bits are set ("1")/reset ("0") according to the condition specified by an instruction.

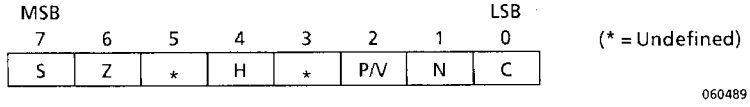


Figure 3.2 Flag Register Configuration

The following 4 bits are directly available to the programmer for setting the jump, call and return instruction conditions:

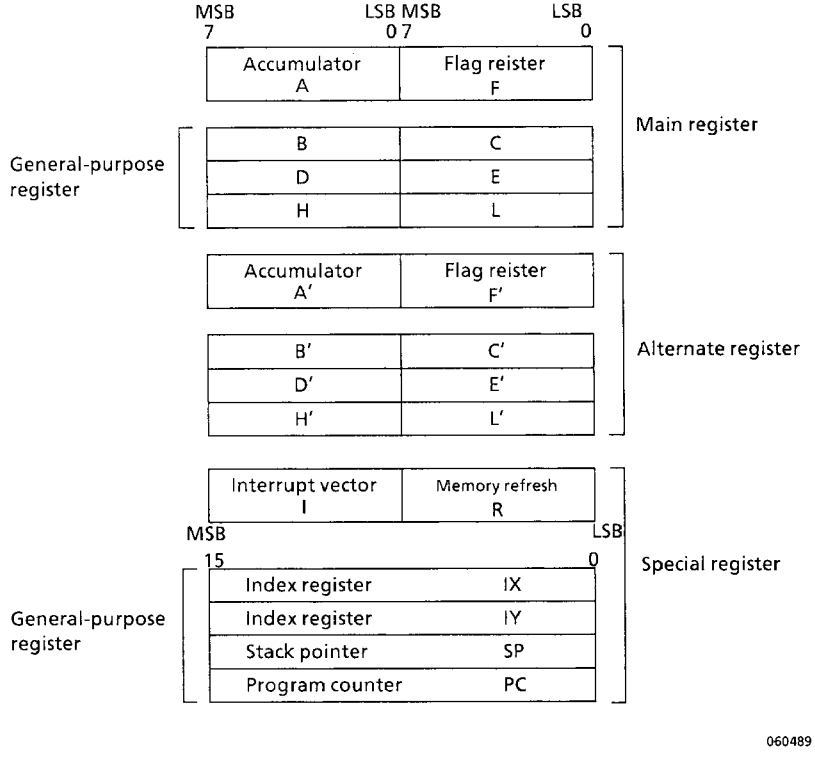


Figure 3.3 Flag Register Configuration

- Sign flag (S)

  When the result of an operation is negative, the S flag is set to "1" . Actually, the content of bit 7 of accumulator is stored in this flag.

- Zero flag (Z)

  When all bits turn out to be "0" s after operation, the Z flag is set to "1" . Otherwise, it is set to "0".

With a block search instruction (CPI, CPIR, CPD or CPDR), the Z flag is set to "1" if the source data and the accumulator data match.

With a block I/O instruction (INI, IND, OUTI or OUTD), the Z flag is set to "1" if the content of the B register used as the byte counter is "0" at the end of comparison.

- Parity/overflow flag (P/V)

This flag has two functions. One is the parity flag (P) that indicates the result of a logical operation (AND A, B etc.). The P flag is set to "1" if the parity is even as a result of the operation on signed values by two's complement. It is reset to "0" if the parity is odd. With a block search instruction (CPI, CPIR, CPD or CPDR) and a block transfer instruction (LDI or LDD), the P flag indicates the state of the byte counter (register pair B and C). It is set to "1" if the byte counter is not "0" and reset to "0" when the byte counter becomes "0" (at the end of comparison or data transfer). The content of the interrupt enable flip-flop (IFF) is saved to the P flag when the contents of the R register or I register are transferred to the accumulator.

The other use of the P/V flag is the overflow flag (V) that indicates whether an overflow has occurred or not as a result of an arithmetic operation. The V flag is set to "1" when the value in the accumulator gets out of a range of the maximum value +127 and the minimum value −128 and therefore cannot be correctly represented as a two's complement notation.

Whether the P/V flag operates as the P flag or V flag is determined by the type of the instruction executed.

- Carry flag (C)

The C flag is set to "1" if a carry occurs from bit 7 of the accumulator or a borrow occurs as a result of an operation.

The following two flags are not available to the programmer for the test and set ("1")/reset ("0") purposes. They are internally used by the MPU for BCD arithmetic operations.

  - Half carry flag (H)

The H flag is used for holding the carry or borrow from the low-order 4 bits of a BCD operation result. When a DAA instruction (decimal adjust) is executed, the MPU automatically uses the H flag to adjust the result of a decimal addition or subtraction.

  - Add/subtract flag (N)

In BCD operation, algorithm is different between addition and subtraction. The N flag indicates whether the executed operation is addition or subtraction.

For change of the flag state depending on the instruction, see 3.4 "TMPZ84C00A Instruction Set".

(c) General-purpose registers (B, C, D, E, H, L)

General-purpose registers consist of 8 bits each. They are used as 16-bit register pairs (BC, DE, HL) as well as independent 8-bit registers to supplement the accumulator. The B register and the register pair BC are used as counters when a block I/O, block transfer, or search instruction is executed. The register pair HL has various memory addressing features as compared with the register pairs BC and DE.

(2) Alternate registers (A', F', B', C', D', E', H', L')

The configuration of the alternate registers is exactly the same as that of the main registers. There is no instruction that handles the alternate registers directly. The data in the alternate registers are processed by moving them into the main registers by means of exchange instructions as shown below:

EX AF, AF' (A↔A', F↔F')

EXX (B↔B', C↔C', D↔D', E↔E', H↔H', L↔L')

When a high-speed interrupt response has been requested within the system, these instruction can be used to quickly move the contents of the accumulator, flag registers, and general-purpose registers into the corresponding registers. This eliminates the need for transferring the register contents to/from the external stack during execution of the interrupt handling routine, thereby shortening the interrupt servicing time greatly.

(3) Special purpose registers (I, R, IX, IY, SP, PC)

(a) Interrupt page address register (I)

The TMPZ84C00A provides two kinds of interrupts :maskable interrupt (INT) and non-maskable interrupt (NMI). The maskable interrupt provides three modes (0, 1, and 2) in which the interrupt is handled. These modes can be selected by instructions IM0, IM1, and IM2 respectively. In Mode 2, any memory location can be called indirectly depending on the interrupt. For this purpose, the I register stores the high-order 8 bits of the indirect address. The low-order 8 bits are supplied from the interrupting peripheral LSI. This scheme permits calling the interrupt handling routine from any memory location in an extremely short access time. For the details of interrupts, see [4] "Interrupt Capability".

(b) Memory refresh register (R)

The R register is used as the memory refresh counter when the dynamic RAM is used for memory. This permits using of the dynamic memory in the same manner as the static memory. The Low-order 7 bits of this 8-bit register is automatically incremented for each instruction fetch. While the MPU decodes and executes the fetched instruction, the contents of the R register are synchronized with the refresh signal to place the low-order 8 bits on the address bus. This operation is all performed by the MPU and, therefore, dose not need a special processing by program. The MPU operation is not delayed by this operation. During refresh, the contents of the I register are placed on the high-order 8 bits of the address bus.

(c) Index registers (IX, IY)

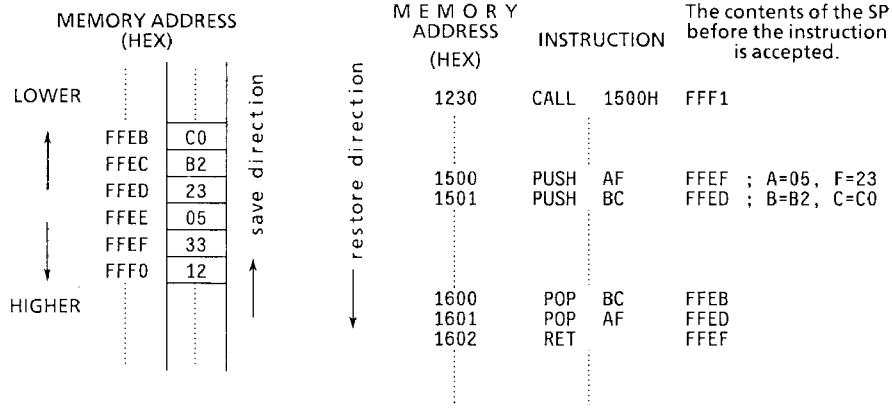
The two independent index registers IX and IY hold the 16-bit base address when used in the index addressing mode. In this addressing mode, the memory address obtained by adding the contents of an index register to the displacement value (for example, LD IX+40H) is specified. This mode is convenient for using data tables. Also these registers can be used separately for memory addressing and data retaining registers.

(d) Stack pointer (SP)

The stack pointer is a 16-bit register to provide the start address information in the stack area in the external RAM. The content of the stack pointer is decremented at the execution of a CALL instruction or PUSH instruction or interrupt handling and is incremented at the execution of a return instruction or POP instruction. At the execution of a CALL instruction or interrupt handling, the current content of the program counter is saved into the stack. At the execution of a return instruction, the content is restored from the stack to the program counter. These operations are all performed by the MPU automatically. However, the other registers are not saved or restored automatically. For the storing of the contents of these registers, an exchange instruction (EX or EXX) for alternate register, a PUSH or a POP instructions must be used. When a PUSH instruction is executed, the contents of the specified register are saved into the stack. When a POP instruction is executed, the contents of the stack are moved to the specified register.

These data are restored on a last-in, first-out basis. Use of the stack permits processing of multiple-level interrupts, deep subroutine nestings, and various data manipulation very easily. The stack pointer is not initialized in the hardware approach. Therefore, it is required to allocate the stack area in RAM to specify initialization (at the highest address of the stack area) in the initial program.

(ex)



The foregoing example shows the stack pointer and stack operations in which the instructions starting with the CALL at address 1230H and ending with the RET at address 1602H have been executed. However, it is assumed that there is no instruction or interrupt other than shown above that uses the stack during the execution. When the value the stack pointer before executing the CALL instruction at address 1230H indicates address FFF1H, address 1233H is stored at addresses FFF0H and FFEFH because the CALL instruction consists of 3 bytes, then the stack pointer is decremented. Similarly, the data are saved or restored sequentially according to the instructions. These stack and stack pointer operations are all performed automatically.

(e) Program counter (PC)

The program counter holds, in 16 bits, the memory address of the instruction to be executed next. The MPU fetches the instruction from the memory location indicated by the program counter. When the content of the program counter is put on the address bus, the program counter is incremented automatically. However, it is not incremented with a jump instruction, a call instruction, or interrupt processing. Instead, the specified new address is set on it. With a return instruction, the content restored from the stack is set on the program counter. These operations are all performed automatically and therefore, no care is required for programming.

[2] Halt Capability

When a HALT instruction has been executed, the MPU is put in the halt state. The halt capability can be used to halt the MPU against the external interrupts, thereby reducing the power dissipation. In the halt state the states of MPU's internal registers are retained. The halt state is cleared by reset or when an interrupt is accepted. For the details of halt operation, see [3] "Basic Timing".

(1) Halt operation

When a HALT instruction has been executed, the MPU sets the $\overline{\text{HALT}}$ signal to "0" to indicate that the MPU is going to be put in the halt state. Actually, the MPU in the halt state automatically continues executing NOP instructions if there is the system clock input. However, the program counter is not incremented. This keeps the refresh signal generated when the dynamic memory is used. During halt, the MPU's internal states are retained. By using TLCS-Z80's clock generator/controller (TMPZ84C60P or TMPZ84C61AP), the clock input control for these halt operations is realized easily.

(2) Releasing the halt state

The halt state is cleared by accepting an interrupt (the $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ signal input) or by reset (the $\overline{\text{RESET}}$ signal input). When an interrupt is accepted, the halt state is cleared and the interrupt handling routine is executed. However, a maskable interrupt (INT) cannot be accepted unless the interrupt enable flip-flop (IFF) is set.

Note that when the halt state is cleared by the $\overline{\text{RESET}}$ signal, the MPU is reset and the program counter is set to "0".

[3] RESET Signal

Holding the $\overline{\text{RESET}}$ pin at the low level ("0") under the following conditions, the MPU's internal states are reset:

(1) The supply voltage level is within the operational voltage range.

(2) System clock stabilization.

(3) Holding the $\overline{\text{RESET}}$ signal at the low level ("0") for at least 3 full clock cycles. When the $\overline{\text{RESET}}$ signal goes high ("1"), the MPU starts executing instructions from address 0000H after at least 2T state dummy cycles.
When reset, the MPU performs the following processing:

(1) Program counter

0000H is set.

(2) Interrupt

The interrupt enable flip-flop (IFF) is reset to "0" to disable the maskable interrupt. For the maskable interrupt processing, mode 0 is specified.

(3) Control output

All control outputs are made inactive ("1"). Therefore, the halt state is also cleared.

(4) Interrupt page address register (I register)

The content of the R register becomes 00H.

(5) Refresh register (R register)

The content of the R register becomes 00H.

The contents of the registers other than above and the external memory do not change.

Therefore, they must be initialized as required.

[4] Interrupt Capability

The interrupt capability is used to suspend the execution of the current program and execute the processing of the requested peripheral LSI. Normally, this interrupt processing routine contains the data exchange and transfer of status and control information between the MPU and the peripheral LSI. When this routine has been completed, the MPU returns to the active state before the interrupt has been accepted.

The TMPZ84C00A provides the non-maskable interrupt (NMI) and maskable interrupt (INT) capabilities which are detected by the $\overline{NMI}$ and $\overline{INT}$ interrupt request signals, respectively. A non-maskable interrupt, when caused by a peripheral LSI, is accepted unconditionally. This interrupt is used to support critical functions such as the protection of the system from unpredictable happening including power failure. A maskable interrupt can be enabled or disabled by program. For example, if the timer is used and, therefore, an interrupt is not desired, the system can be programmed to disable the interrupt. Table 3.1 lists the processing by interrupt source.

(1) Interrupt enable/disable

A non-maskable interrupt cannot be disabled by program, while a maskable interrupt can be enabled or disabled by program. The MPU has the interrupt enable flip-flop (IFF). A maskable interrupt can be enabled or disabled by setting this flip-flop to "1" (set) or "0" (reset) through an EI instruction (enable) or a DI instruction (disable) in program. Actually, the IFF consists of two flip-flops IFF1 and IFF2. IFF1 is used to select between the enable and disable of a maskable interrupt. IFF2 holds the state of IFF1 before a maskable interrupt has been accepted. Both IFF1 and IFF2 are reset to "0" when any of the following conditions occurs, disabling an interrupt:

- MPU reset
- Execution of DI instruction
- Acceptance of maskable interrupt

Both IFF1 and IFF2 are set to "1" when the the following condition occurs, enabling an interrupt:

- Execution of EI instruction

Actually, the waiting maskable interrupt request is accepted after the execution of the instruction that follows the EI instruction.
This delay by one instruction is caused by accepting an interrupt after completion of the execution of a return instruction if the instruction following the EI instruction is a return instruction.
In the above operation, the contents of IFF1 and IFF2 are the same.

Table 3.1 Processing by Interrupt Source

Interrupt Source	Priority	Programmed condition		Vector address	Interrupt return instruction
Non-maskable interrupt (the falling edge of $\overline{NMI}$)	1	None		Address 66H	RETN
Maskable interrupt ($\overline{INT}$ becomes "0" at instruction's last clock)	2	IFF = 1	Mode 0	Instruction from peripheral LSI. Normally, CALL or RST instruction.	(Note) RET I
			Mode 1	Address 38H.	
			Mode 2	The address indicated by the data table (memory) at the address specified by I register (high-order 8 bits) and data from peripheral LSI (low-order 8 bits, LSB = "0").	

060489

Note : Mode 0 applies when the instruction from peripheral LSI is CALL or RST instruction.

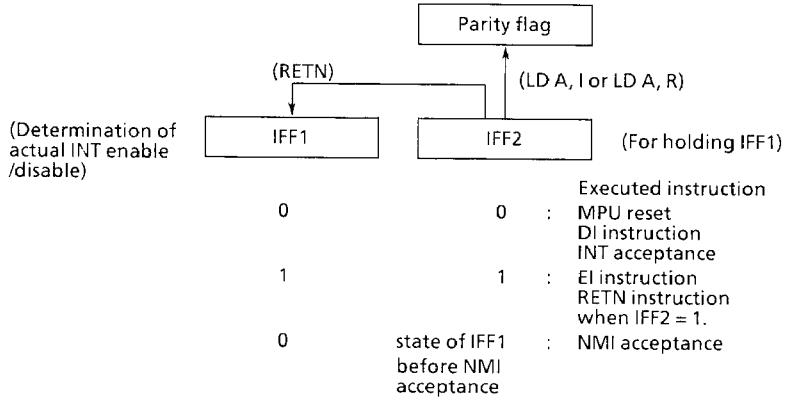


Figure 3.4 Interrupt Enable Flip-Flop (IFF)

When a non-maskable interrupt has been accepted, IFF1 is reset to "0" (interrupt disable) until an EI or RETN instruction is executed, so as to prevent from accepting the next interrupt. For this purpose, the state (interrupt enable/disable) of IFF1 immediately before non-maskable interrupt acceptance must be stored. This state is copied into IFF2 upon acceptance of a non-maskable interrupt. The content of IFF2 is copied into the parity flag at the execution of the following instructions, so that the copied data can be tested or stored:

- The load instruction (LD A, I) to load the contents of the I register into the accumulator.
- The load instruction (LD A, R) to load the contents of the R register into the accumulator.

When the return instruction (RETN) from the non-maskable interrupt is executed, the contents of the current IFF2 are copied back to IFF1. If an operation which changes the contents of IFF2 (due to the execution of EI or DI instruction, for example) has not been performed during interrupt handling, IFF1 automatically returns to the state immediately before the interrupt acceptance. Table 3.2 lists the states of IFF1 and IFF2 after execution of interrupt-related instructions.

Table 3.2 State of IFF1 and IFF2

Operation sequence	IFF1	IFF2	Remarks
NPU reset	0	0	
EI	1	1	
NMI acceptance	0	1	
LD A, I	*	*	Parity flag←IFF2
RETN	1	1	IFF1←IFF2
LD A, R	*	*	Parity flag←IFF2
INT acceptance	0	0	
RETI	*	*	
EI	1	1	
NMI acceptance	0	1	
DI	0	0	
RETN	*	*	

Note : * = no change 060489

(2) Interrupt processing

With a non-maskable interrupt, the internal NMI flip-flop is set to "1" on the falling edge of the interrupt signal, $\overline{NMI}$. The state of this flip-flop is sampled on the rising edge of the last clock of each instruction to accept an interrupt. A maskable interrupt is accepted if the interrupt signal $\overline{INT}$ is low ("0") on the rising edge of the last clock of each instruction and the interrupt enable state (IFF = 1 and $\overline{BUSREQ}$ signal = inactive ("1")) is on. The following is the processing to be performed after a non-maskable interrupt and a maskable interrupt are accepted:

(a) Non-maskable interrupt (NMI)

When a non-maskable interrupt has been accepted, the MPU performs the following processing:

1 The internal NMI flip-flop is reset to "0".

2 IFF1 is reset to "0", disabling the maskable interrupt.

The contents of the IFF1 immediately before the interrupt acceptance are copied into the IFF2.

3 The contents of the current program counter are saved into the stack.

4 The instructions starting from non-maskable interrupt vector address 66H are executed.

A non-maskable interrupt processing program terminates after executing the RETN instruction. This return instruction performs the followings:

1 The contents of the current IFF2 are copied into IFF1.

2 The contents of the program counter are restored from the stack.

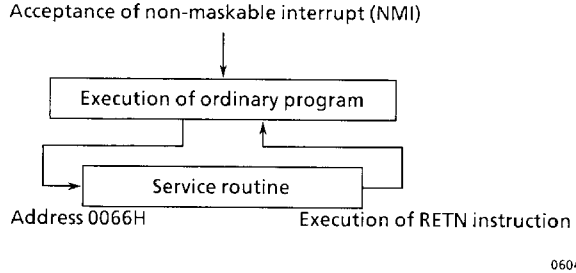


Figure 3.5 Non-Maskable Interrupt Processing

(b) Maskable interrupt (INT)

When a maskable interrupt has been accepted, the MPU performs the following processings:

1 Both IFF1 and IFF2 are reset to "0", disabling the maskable interrupts.

2 The contents of the current program counter are saved into the stack.

3 A maskable interrupt is serviced in one of the three modes 0, 1 and 2. A mode is selected by executing the instruction IM0, IM1or IM2 before the interrupt is serviced. The instructions are executed starting from the vector address corresponding to the selected mode.

- Mode 0

In mode 0, the interrupting peripheral LSI puts a restart instruction (RST) or a call instruction (CALL) on the data bus and the MPU executes the interrupt service routine according to that instruction. At reset, this mode is automatically set.

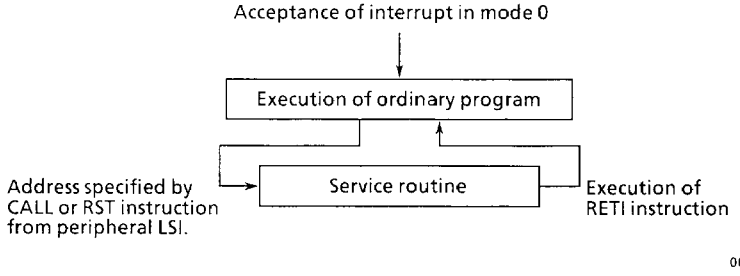


Figure 3.6 Interrupt Processing in Mode 0

- Mode 1

When an interrupt is accepted in mode 1, restart is performed from address 0038H. Therefore, the service routine for this interrupt is programmed from the address 0038H .

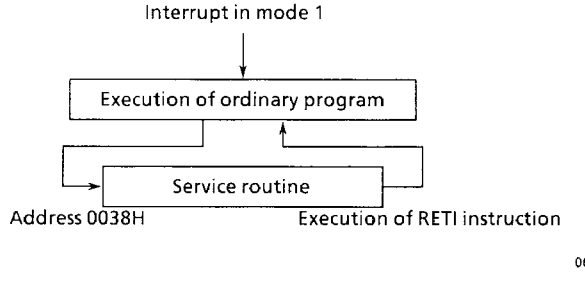


Figure 3.7 Interrupt Processing in Mode 1

- Mode 2

The interrupt processing in mode 2 requires a 16-bit pointer consisting of the high-order 8 bits of the I register and the low-order 8 bits (with the LSB="0") of the data fetched from the TLCS-Z80 family peripheral LSI. Therefore, the necessary value must be loaded in the I register beforehand. This pointer is used to specify the memory address in the table. The contents of the specified address and the next address provide the start address of the service routine. Therefore, use of this mode requires the table of the service routine's start address (16 bits) to be set at appropriate location under software control . This location can be anywhere in memory.

The LSB of the table pointer is set to "0" because a 2-byte data is needed to specify the service routine start address in 16 bits and start that address from an even-number address. In the table, the start address begins with the low-order byte followed by the high-order byte as shown in Figure 3.8.

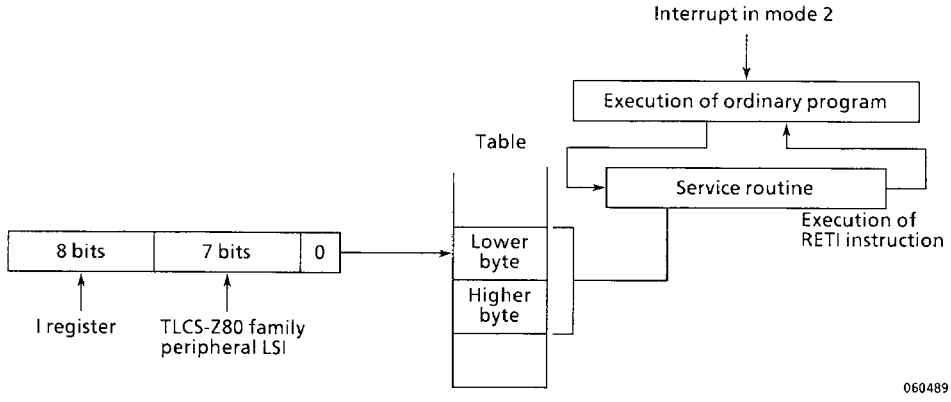


Figure 3.8 Interrupt Processing in Mode 2

Mode 2 is used in the daisy chain interrupt processing using TLCS-Z80 family LSI. TLCS-Z80 family peripheral LSIs all contain the interrupt priority controller in daisy chain structure. In this interrupt structure, the interrupt request signals are connected one after another and given priorities for processing when two or more maskable interrupt requests occur at a time. Only the interrupt vector from the peripheral LSI having the highest priority is put on the data bus. By receiving the interrupt vector in mode 2, the processing for that peripheral LSI can be performed. When an interrupt requested by a peripheral LSI having a priority higher than that of the current peripheral LSI during the execution of the interrupt processing routine, the higher priority interrupt can be enabled by the EI instruction to form an interrupt nesting.

The maskable interrupt processing program terminates by executing an RETI instruction. This return instruction performs the following processings:

- Restores the content of the program counter from the stack.
- Notifies the requesting peripheral LSI of the termination of interrupt processing.

3.3 MPU STATUS TRANSITION DIAGRAM AND BASIC TIMING

The following describes the MPU status transition and the basic timing of each MPU operation.

[1] Instruction Cycle

Each TMPZ84C00A instruction is executed by combining the basic operations of memory read/write, input/output, bus request/acknowledge, and interrupt. These basic operations are performed synchronizing with the system clock (the CLK signal).

One clock period is called a state (T). The smallest unit of each basic operation is called a machine cycle (M). Each instruction consists of 1 to 6 machine cycles and each machine cycle consists of 3 to 6 clock states basically. However, the number of clock states in a machine cycle can be increased by the $\overline{WAIT}$ signal described later on. Figure 3.9 shows an example of the basic timing of a 3-machine-cycle instruction.

The first machine cycle (M1) of each instruction is the cycle in which the Opcode of the instruction to be executed next is read (this is called the Opcode fetch cycle). The Opcode fetch cycle basically consists of 4 to 6 clock states. In the machine cycle that follows the Opcode fetch cycle, data is transferred between the MPU and the memory or peripheral LSIs. This operation basically consists of 3 to 5 clock states.

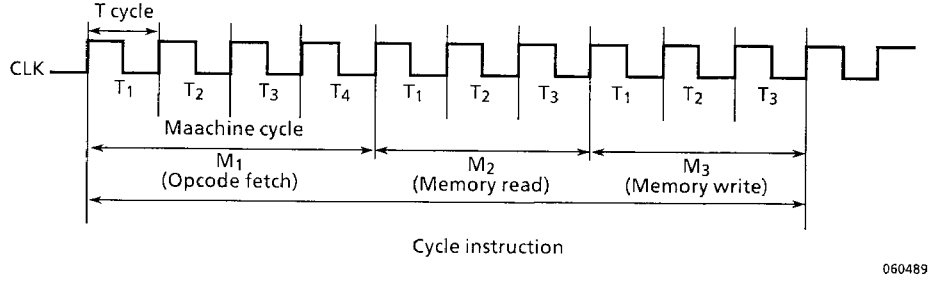


Figure 3.9 Example of MPU Basic Timing (3-Machine-Cycle Instruction)

[2] Status Transition Diagram

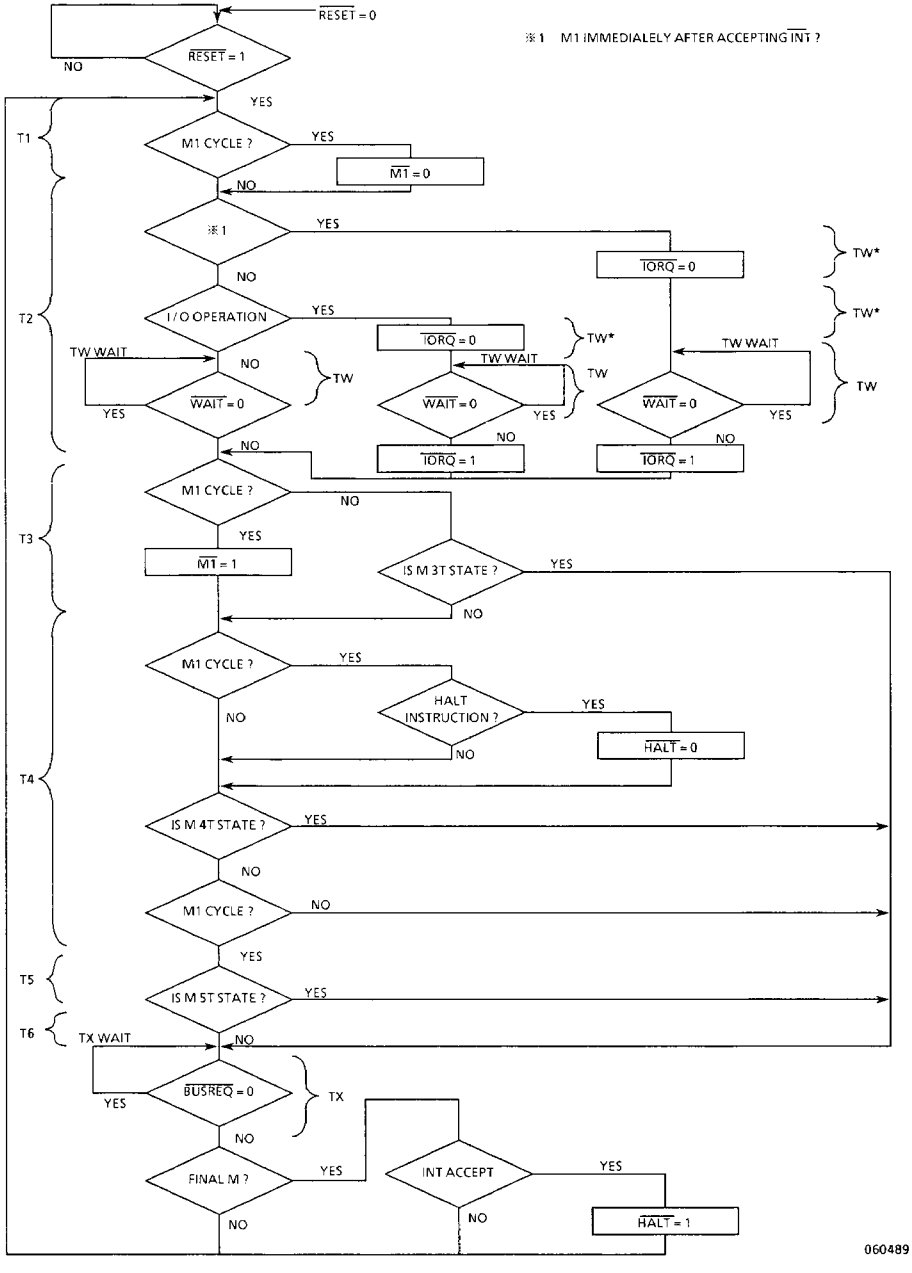


Figure 3.10 Status Transition Diagram

[3] Basic Timing

(1) Opcode fetch cycle (M1)

In the Opcode fetch cycle, MPU fetches an Opcode in the machine-language codes in memory. This is also called the M1 cycle because it is the first machine cycle to execute each instruction.

Figure 3.11 shows the basic timing of a basic Opcode fetch cycle.

In clock state T1, the content of the program counter is put on the address bus. The $\overline{M1}$ signal goes "0", indicating to the MPU that this is the Opcode fetch cycle. At the same time, $\overline{MREQ}$ and $\overline{RD}$ signals go "0". When the $\overline{MREQ}$ signal goes "0", the address signal has already been stabilized. Therefore, this signal can be used for the memory chip enable signal. The $\overline{RD}$ signal indicates that the MPU is ready to accept the data from memory. By these signals, the MPU accesses memory to fetch the Opcode in the instruction register. The MPU samples the $\overline{WAIT}$ signal on the falling edge of clock state T2. If the $\overline{WAIT}$ signal is "0" on the falling edge of clock state T2 and the following wait state (TW), the next state becomes clock state TW. Figure 3.12 shows the delay state of the Opcode fetch cycle caused by the $\overline{WAIT}$ signal.

The data (Opcode) on the data bus is fetched on the rising edge of clock state T3 then, the $\overline{MREQ}$, $\overline{RD}$, and $\overline{M1}$ signals go "1". In clock state T3, a memory refresh address is put on the low-order 8 bits of the address bus and the $\overline{RFSH}$ signal goes "0" and the $\overline{MREQ}$ signal goes "0" again. This signal indicates that the memory refresh cycle is on. At this time, the contents of the I register are put on the high-order 8 bits of the address bus and the 8 bits of the R register are put on the low-order 8 bits of the address bus. By using the $\overline{RFSH}$ and $\overline{MREQ}$ signals, memory refresh is performed in clock state T3 and T4. However, the $\overline{RD}$ signal remains "1" because the contents of the memory refresh address are not put on the data bus.

In clock state T4, the $\overline{MREQ}$ signal returns to "1". The refresh address is kept output until the rising edge of the clock state T1 in the next machine cycle, keeping the $\overline{RFSH}$ signal set to "0". The cycle delay state caused by setting the $\overline{WAIT}$ signal to "0" is the same in the memory read/write, input/output, and maskable interrupt acknowledge cycles. The diagram of the cycle delay state caused by the $\overline{WAIT}$ signal set to "0" is omitted in the following description.

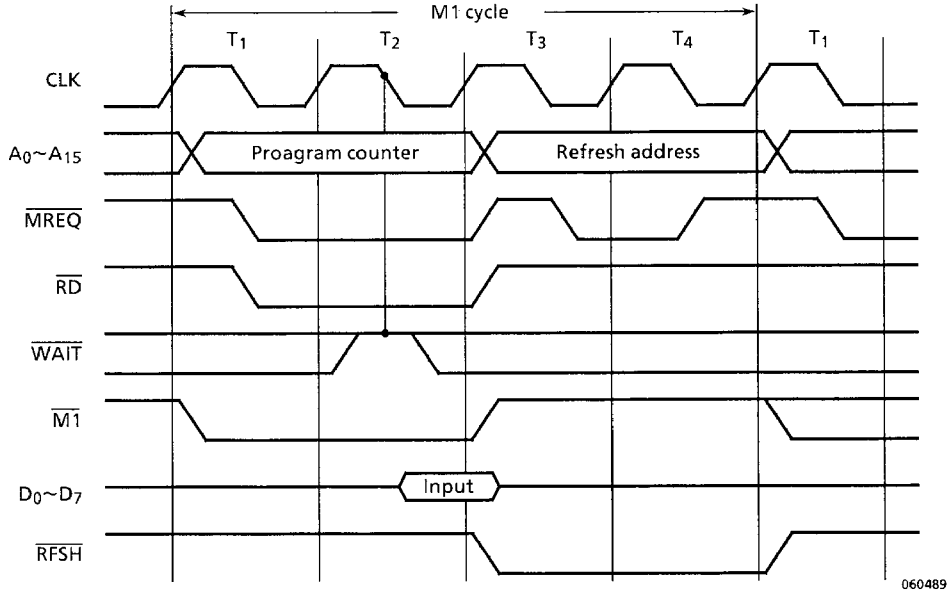


Figure 3.11 Opcode Fetch Timing

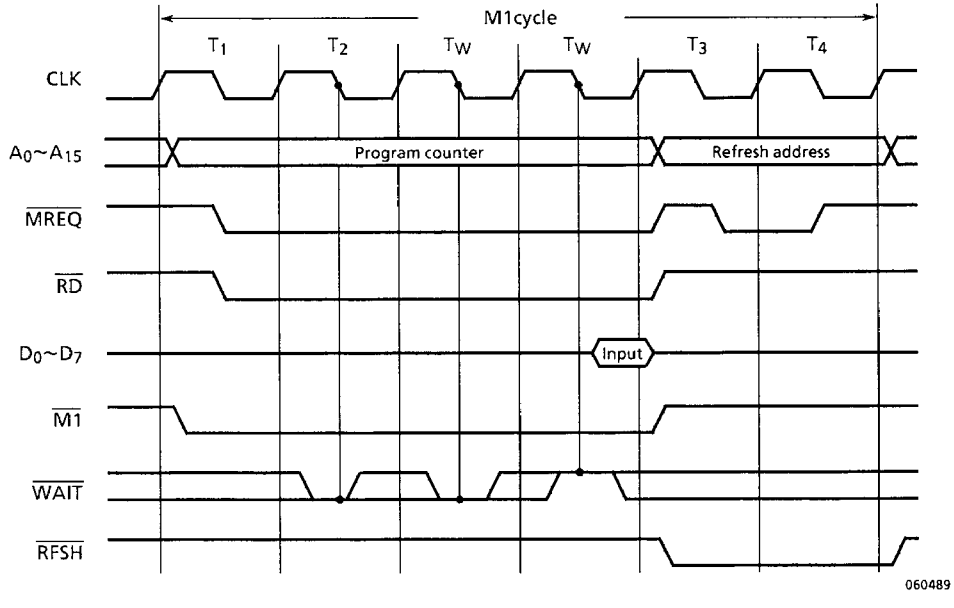


Figure 3.12 Opcode Fetch Timing Incliding Wait State

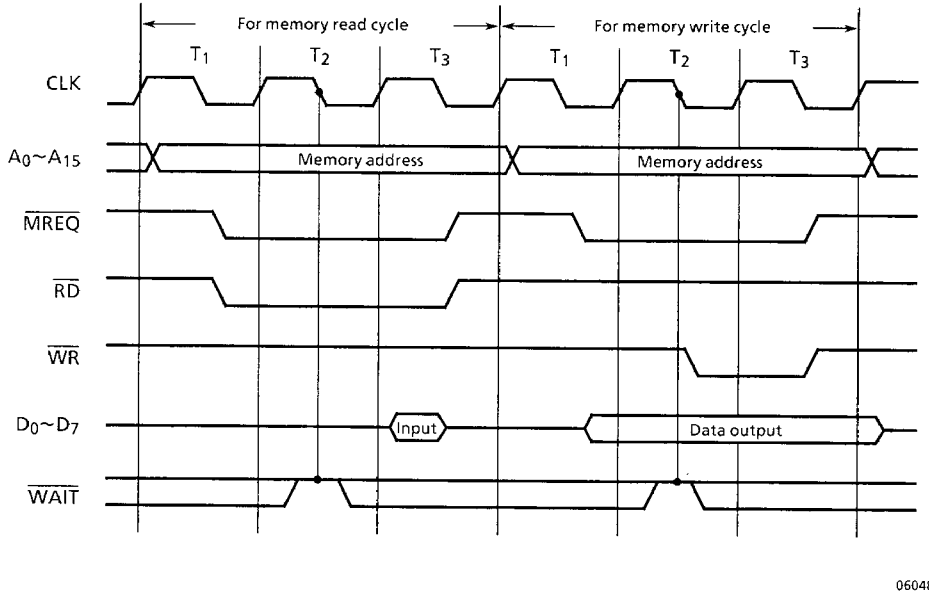
(2) Memory read/write operations

Figure 3.13 shows the basic timing of memory read/write operations (except for the Opcode fetch cycle) in the same diagram for convenience.

In each operation, the memory address signal to read/write data on the address bus is output in clock clock state T1. The operation in which the $\overline{WAIT}$ signal is sampled in clock state T2 and the following TW state is the same as the Opcode fetch cycle.

In memory read, memory data is put on the data bus by the address, $\overline{MREQ}$, and $\overline{RD}$ signals. The MPU reads this data.

In memory write, the memory address signal is put on the address bus then the $\overline{MREQ}$ signal is set to "0" to put the write data onto the data bus. When the data bus has been stabilized, the $\overline{WR}$ signal is output in clock state T2. The WR signal can be used as the memory write signal.



060489

Figure 3.13 Memory Read/Write Cycle Timing

(3) Input/output operations

Figure 3.14 shows the basic timing of input/output operations. The feature of the I/O operation timing is that, regardless of the state of the $\overline{WAIT}$ signal in clock state T2, the I/O cycle automatically goes in the wait state (TW*) after clock T2. The $\overline{WAIT}$ signal is sampled on the falling edge of TW*. If the $\overline{WAIT}$ signal is "0" on the falling edges of TW* and the following clock state, the I/O operation enters into clock state TW*. Clock state TW* is inserted because the $\overline{IORQ}$ signal goes "0" in clock state T2, so that it is too late to sample the $\overline{WAIT}$ signal after decoding the I/O port address. In each of input and output operations, the I/O port address is put on the low-order 8 bits of the address bus in clock state T1. On the high-order 8 bits, the contents of the accumulator or B register are output. In clock state T2, the $\overline{IORQ}$ signal goes "0" instead of the $\overline{MREQ}$ signal. The $\overline{IORQ}$ signal can be used as the chip enable signal for a peripheral LSI.

In an input operation, the contents of the input port are read onto the data bus by the address, $\overline{IORQ}$, or $\overline{RD}$ signals. The MPU reads this data.

In an output operation, the output port address and the output data are respectively put on the address bus and data bus in clock state Tl, then the $\overline{IORQ}$ and $\overline{WR}$ signals go "0" in clock state T2. The $\overline{WR}$ signal can be used as the output port write signal.

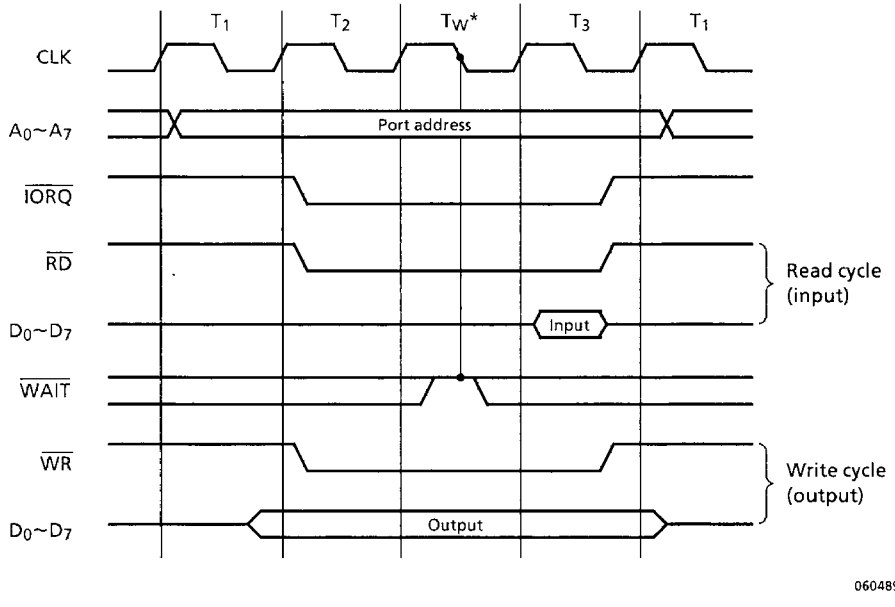


Figure 3.14 I/O Operating Timing

(4) Bus request and bus acknowledge operations

Figure 3.15 shows the basic timing of bus request and bus acknowledge operations.

The address bus (A0 through A15), data bus (D0 through D7), $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ signals controlled by the MPU can be put in the high-impedance state (floating) to electrically disconnect them from the MPU. This operation, after sampling the $\overline{BUSREQ}$ signal on the rising edge of the last clock of each machine cycle, starts on the rising edge of the next clock if this signal is found "0".

Subsequently, these buses are controlled by external peripheral LSIs. For example, data can be directly transferred between memory and these peripheral LSIs. This state is cleared if the $\overline{BUSREQ}$ signal is found "1" after sampling it on the rising edge of each subsequent clock state (TX), and enters into the next machine cycle. During the floating state, the $\overline{BUSACK}$ signal goes "0" to indicate it to the peripheral LSIs.

In this state, however, no memory refresh is performed and, therefore, the $\overline{RFSH}$ signal is set to "1". Hence, to maintain this state for a long time with a system using dynamic memory, memory refresh must be performed by the external controller.

Note that, in the floating state, neither maskable interrupt (INT) nor non-maskable interrupt (NMI) can be accepted.

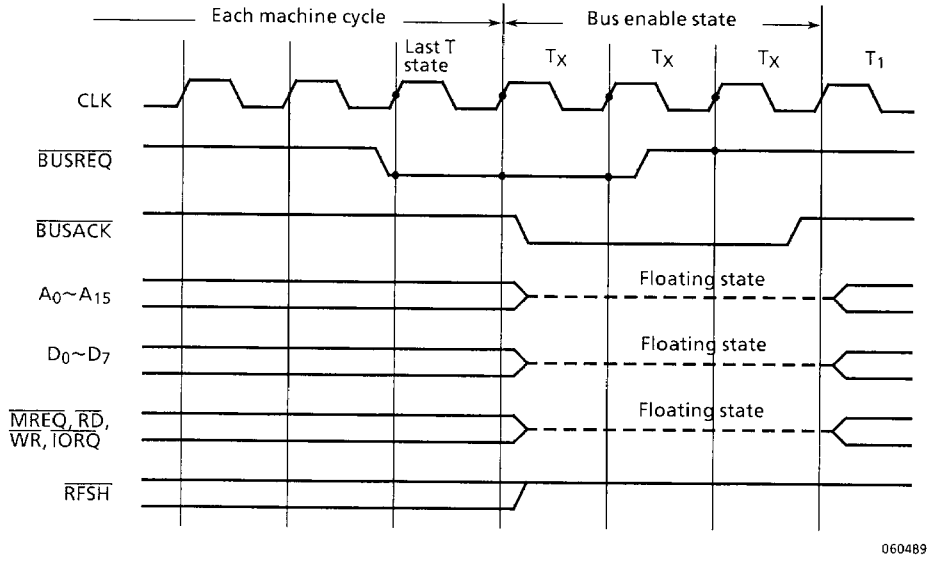


Figure 3.15 Bus Request and Bus Acknowledge Timing

(5) Maskable interrupt acknowledge operation

Figure 3.16 shows the basic timing of the maskable interrupt acknowledge.

The MPU samples the maskable interrupt request signal ($\overline{INT}$) on the rising edge of the last clock of each instruction execution. If the $\overline{INT}$ signal is found "0", a maskable interrupt is accepted except in the following cases:

- The interrupt enable flip-flop is reset to "0".
- The $\overline{BUSREQ}$ signal is "0".

When a maskable interrupt has been accepted, a special Opcode fetch cycle is generated. In this cycle, 2 clock states of wait state (TW*) is automatically inserted after the clock state T2. The $\overline{WAIT}$ signal is sampled on the falling edges of the second clock state TW* and the following clock state TW and, if the $\overline{WAIT}$ signal is found "0", the instruction cycle enters in the next clock state TW. In this Opcode fetch cycle, the $\overline{IORQ}$ signal goes "0" in the first TW* state instead of the $\overline{MREQ}$ signal while, in a normal Opcode fetch cycle, the $\overline{MREQ}$ signal goes "0" in clock state T1. This indicates to the maskable interrupt requesting LSI that the 8-bit interrupt vector can be put on the data bus. The MPU reads this data to perform interrupt processing. Therefore, the contents of the program counter put on the address bus are not used. Unlike an ordinary I/O operation, the $\overline{RD}$ signal does not go "0".

In clock state T3, the memory refresh address signal is put on the address bus for memory refresh like normal Opcode fetch cycle and the $\overline{RFSH}$ signal goes "0". In the subsequent machine cycles (M2 and M3) , the contents of the current program counter are saved into the stack. In machine cycles M4 and M5, the contents of the I register (the high-order 8 bits) and the contents of the address indicated by the address of the vector (the low-order 8 bits) from the peripheral LSI are fetched in the program counter.

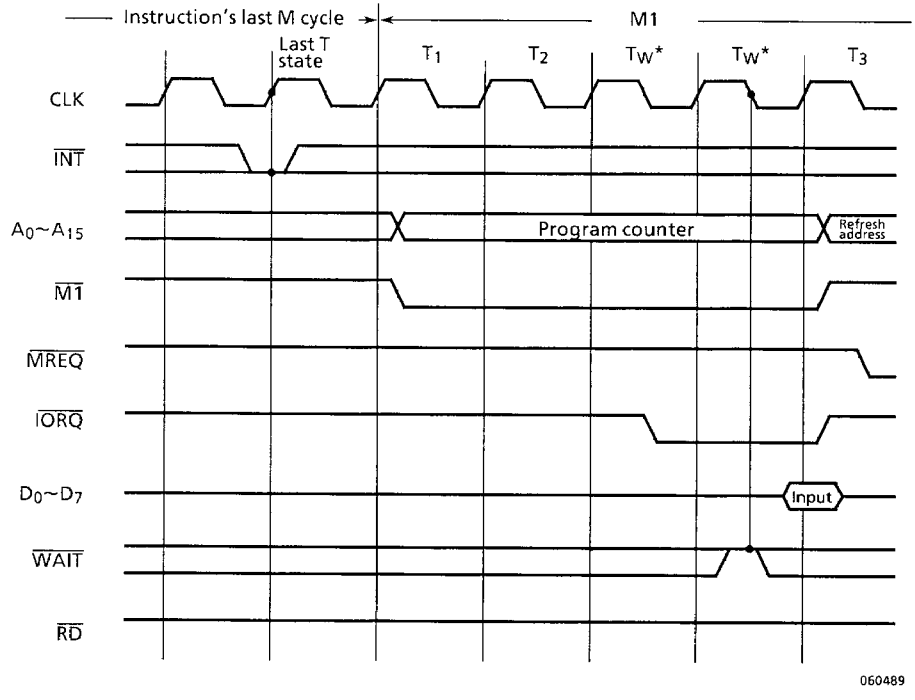


Figure 3.16 Maskable Interrupt Acknowledge Timing

(6) Non-maskable interrupt acknowledge operation

Figure 3.17 shows the basic timing of non-maskable interrupt acknowledge.

When the non-maskable interrupt request signal ($\overline{NMI}$) goes low, the internal non-maskable flip-flop is set to "1". The $\overline{NMI}$ signal is detected in any timing of each instruction. However, the internal NMI flip-flop is sampled on the rising edge of the last clock of each instruction. Therefore, the $\overline{NMI}$ signal should go low by the last clock state of an instruction.

The Opcode fetch cycle for non-maskable interrupt request acknowledge is generally the same as the ordinary Opcode fetch cycle. However, the Opcode on the data bus at the time is ignored. The contents of the current program counter are saved into the stack in the subsequent machine cycles (M2 and M3) . In the following machine cycle, the operation jumps to address 0066H, the non-maskable interrupt vector address. The machine cycles after these depend on the contents of the fetched Opcode.

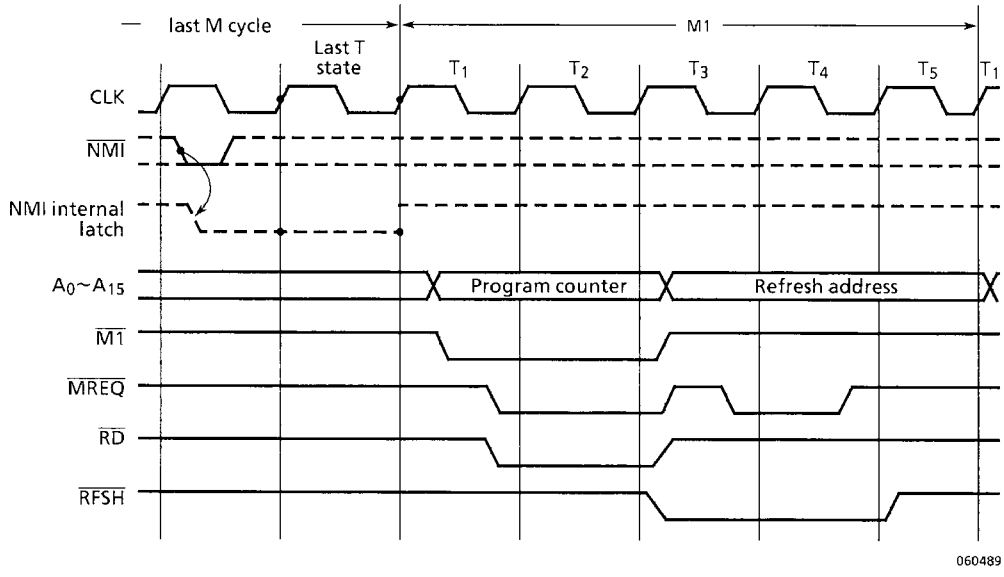


Figure 3.17 Non-Maaskable Inpterrupt Acknowledge Timing

(7) Halt operation

When a HALT instruction is fetched in the Opcode fetch cycle, the MPU sets the $\overline{HALT}$ signal to "0" synchronized with the falling edge of clock state T4 to indicate it to the peripheral LSI and stops operating. If the system clock is kept supplied in the halt state, the MPU continues executing NOP instructions. This is done to output refresh signals when the dynamic memory is used. The NOP instruction execution cycle is the same as the ordinary Opcode fetch cycle except the data on the data bus are ignored.

The halt state is cleared when an interrupt is accepted or the $\overline{RESET}$ signal is set to "0" to reset the MPU. Figure 3.18 shows the halt state clear operation by interrupt acknowledge. An interrupt is sampled on the rising edge of the last clock (clock state T4) of the NOP instruction. A maskable interrupt can be accepted when the $\overline{INT}$ signal is "0". A non-maskable interrupt is accepted when the internal NMI flip-flop which is set on the falling edge of the $\overline{NMI}$ signal is set at "1". However, it is required that the interrupt enable flip-flop is set to "1" for a maskable interrupt to be accepted. The interrupt processing for the accepted interrupt starts from the next cycle.

However, when the supply of the system clock has been stopped by the power down operation, it is required to restart the supply of the system clock and input the $\overline{INT}$ signal until the execution of one instruction is completed or the $\overline{RESET}$ signal until 3 clocks are input. Figure 3.19 shows the timing of clearing the halt state caused by power down. By using TLCS-Z80's clock generator/controller (TMPZ84C60P or TMPZ84C61AP), above-stated operation is realized easily.

For the reset operation, see (8) "Reset operation". Note that the $\overline{INT}$ and $\overline{NMI}$ signals are shown on the same diagram in Figures 3.18 and 3.19 for convenience.

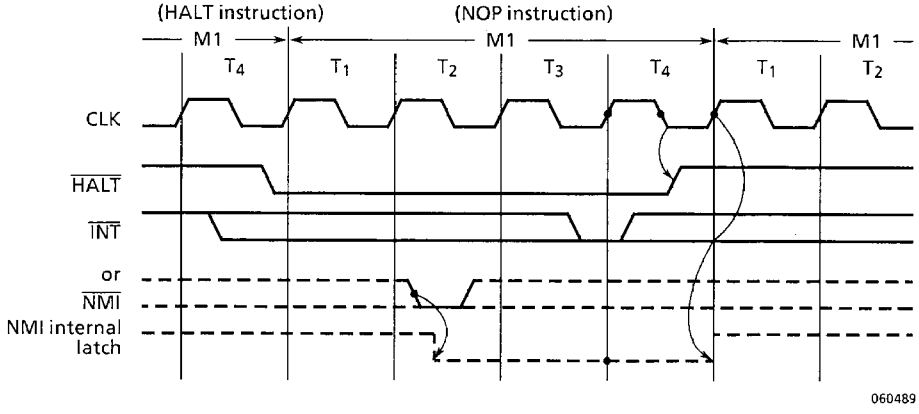


Figure 3.18 Timing of Clearing Halt State Caused by Interrupt Acknowledge

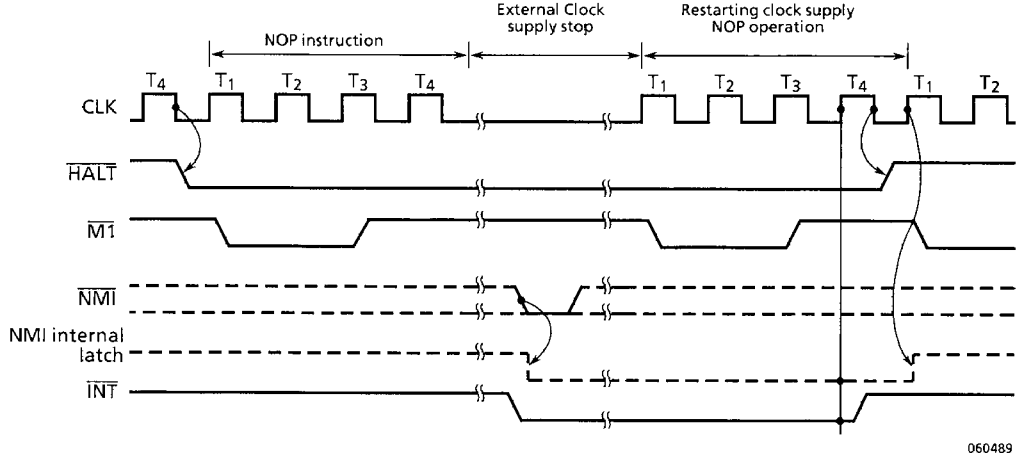


Figure 3.19 Timing of Clearing Halt State Caused By Power Down

(8) Reset operation

Figure 3.20 shows the basic timing of reset operation.

To reset the MPU, the $\overline{\text{RESET}}$ signal must be kept at "0" for at least 3 clocks. When the $\overline{\text{RESET}}$ signal goes "1", instruction execution starts from address 0000H after a dummy cycle of at least 2 clock states.

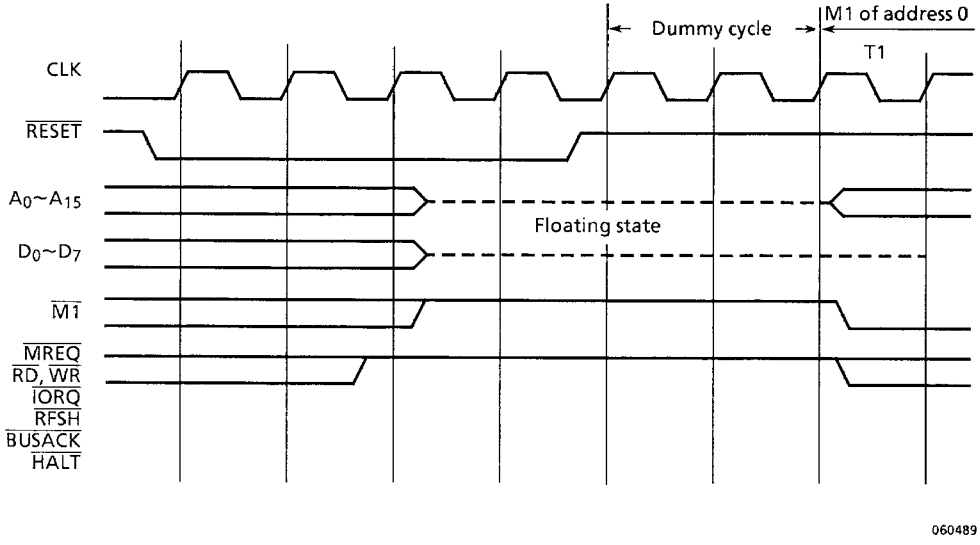


Figure 3.20 Reset Timing

3.4 TMPZ84C00A INSTRUCTION SET

This subsection lists the TMPZ84C00A instruction codes and their functions. The table below lists the symbols and abbreviations used to describe the instruction set. The symbols which require special attention are described in the locations in which they appear.

- Symbols (1/2)

Classification	Symbol	Meaning	
Register	r, g	Register	B, C, D, E, H, L, A,
	t	Register pair	BC, DE, HL
		Stack pointer	SP
	q	Register pair	BC, DE, HL, AF
	p	Register pair	BC, DE
		Index register	IX
		Stack pointer	SP
	s	Register pair	BC, DE
		Index register	IY
		Stack pointer	SP
	t_H	Higher register of register pair (B, D, H)	
		Higher 8 bits of stack pointer (SP)	
	q_H	Higher register of register pair (B, D, H, A)	
	IX_H	Higher 8 bits of index register IX	
	IY_H	Higher 8 bits of index register IY	
	PC_H	Higher 8 bits of program counter (PC)	
	t_L	Lower register of register pair (C, E, L)	
		Lower 8 bits of stack pointer (SP)	
	q_L	Lower register of register pair (C, E, L, F)	
	IX_L	Lower 8 bits of index register IX	
	IY_L	Lower 8 bits of index register IY	
	PC_L	Lower 8 bits of program counter (PC)	
	rb	Bit b (0-7) of register (B, C, D, E, H, L, A)	

120489

- Symbols (2/2)

Classification	Symbol	Meaning
Memory	mn	Memory address represented in 16 bits. m indicates higher 8 bits and n, lower 8 bits.
	$(HL)_b$	Bit b (0-7) of the contents of the memory address indicated by register pair HL.
	$(IX+d)_b$	Bit b (0-7) of the contents of the memory address indicated by the value obtained by adding 8-bit data d to the content of index register IX.
	$(IY+d)_b$	Bit b (0-7) of the contents of the memory address indicated by the value obtained by adding 8-bit data d to the content of index register IY.
Flag change symbol	0	Reset to "0" by operation.
	1	Set to "1" by operation.
	-	No change
	*	Affected by operation
	X	Undefined
	P	Handled as parity flag. P = 0: odd parity P = 1: even parity
	V	Handled as overflow flag. V = 0: No overflow V = 1: Overflow
Operator	←	Transfer
	⇔	Exchange
	+	Add
	−	Subtract
	∧	Logical and between bits.
	∨	Logical or between bits.
	⊕	Exclusive or between bits
Others	IFF	Interrupt enable flip-flop
	CY	Carry flag
	Z	Zero flag

120489

TMPZ84C00A Instruction Set (1/9)

ITEM/ CLASSI-FICA-TION	Assembler mnemonic	Object code Binary 76 543 210	Object code Hex	Function	Flag S	Z		H		P/V	N	C	No. OF CY-CLES	No. OF STA-TES
8-BIT DATA LOAD	LD r,g	01 rrr ggg	40+r×8+g	r←g	–	–	X	–	X	–	–	–	1	4
	LD r,n	00 rrr 110 nn nnn nnn	06+r×8 n	r←n	–	–	X	–	X	–	–	–	2	7
	LD r,(HL)	01 rrr 110	46+r×8	r←(HL)	–	–	X	–	X	–	–	–	2	7
	LD r,(IX+d)	11 011 101 01 rrr 110 dd ddd ddd	DD 46+r×8 d	r←(IX+d)	–	–	X	–	X	–	–	–	5	19
	LD r,(IY+d)	11 111 101 01 rrr 110 dd ddd ddd	FD 46+r×8 d	r←(IY+d)	–	–	X	–	X	–	–	–	5	19
	LD (HL),r	01 110 rrr	70+r	(HL)←r	–	–	X	–	X	–	–	–	2	7
	LD (IX+d),r	11 011 101 01 110 rrr dd ddd ddd	DD 70+r d	(IX+d)←r	–	–	X	–	X	–	–	–	5	19
	LD (IY+d),r	11 111 101 01 110 rrr dd ddd ddd	FD 70+r d	(IY+d)←r	–	–	X	–	X	–	–	–	5	19
	LD (HL),n	00 110 110 nn nnn nnn	36 n	(HL)←n	–	–	X	–	X	–	–	–	3	10
	LD (IX+d), n	11 011 101 00 110 110 dd ddd ddd nn nnn nnn	DD 36 d n	(IX+d)←n	–	–	X	–	X	–	–	–	5	19
	LD (IY+d),n	11 111 101 00 110 110 dd ddd ddd nn nnn nnn	FD 36 d n	(IY+d)←n	–	–	X	–	X	–	–	–	5	19
	LD A,(BC)	00 001 010	0A	A←(BC)	–	–	X	–	X	–	–	–	2	7
	LD A,(DE)	00 011 010	1A	A←(DE)	–	–	X	–	X	–	–	–	2	7
	LD A,(mn)	00 111 010 nn nnn nnn mm mmm mmm	3A n m	A←(mn)	–	–	X	–	X	–	–	–	4	13
	LD (BC),A	00 000 010	02	(BC)←A	–	–	X	–	X	–	–	–	2	7
	LD (DE),A	00 010 010	12	(DE)←A	–	–	X	–	X	–	–	–	2	7
	LD (mn),A	00 110 010 nn nnn nnn mm mmm mmm	32 n m	(mn)←A	–	–	X	–	X	–	–	–	4	13
	LD A,I	11 101 101 01 010 111	ED 57	A←I	*	*	X	0	X	IFF	0	–	2	9
	LD A,R	11 101 101 01 011 111	ED 5F	A←R	*	*	X	0	X	IFF	0	–	2	9
	LD I,A	11 101 101 01 000 111	ED 47	I←A	–	–	X	–	X	–	–	–	2	9
	LD R,A	11 101 101 01 001 111	ED 4F	R←A	–	–	X	–	X	–	–	–	2	9
16-BIT DATA LOAD	LD t,mn	00 tt0 001 nn nnn nnn mm mmm mmm	01+t×10 n m	t←mn	–	–	X	–	X	–	–	–	3	10
	LD IX,mn	11 011 101 00 100 001 nn nnn nnn mm mmm mmm	DD 21 n m	IX←mn	–	–	X	–	X	–	–	–	4	14

r	rrr
g	ggg
B	000
C	001
D	010
E	011
H	100
L	101
A	111

t	tt
BC	00
DE	01
HL	10
SP	11

Note : r,g means any of the registers A, B, C, D, E, H, L.
IFF in "Flag" column indicates that the content of the interrupt enable flip-flop is copied into the P/V flag.

120489

TMPZ84C00A Instruction Set (2/9)

ITEM/CLASSIFICATION	Assembler mnemonic	Object code Binary (76 543 210)	Object code Hex	Function	Flag S	Z		H		P/V	N	C	No. OF CYCLES	No. OF STATES
16-BIT DATA LOAD	LD IY,mn	11 111 101 00 100 001 nn nnn nnn mm mmm mmm	FD 21 n m	IY←mn	–	–	X	–	X	–	–	–	4	14
	LD HL,(mn)	00 101 010 nn nnn nnn mm mmm mmm	2A n m	H←(mn+1) L←(mn)	–	–	X	–	X	–	–	–	5	16
	LD t,(mn)	11 101 101 01 tt1 011 nn nnn nnn mm mmm mmm	ED 48+t×10 n m	t_H←(mn+1) t_L←(mn)	–	–	X	–	X	–	–	–	6	20
	LD IX,(mn)	11 011 101 00 101 010 nn nnn nnn mm mmm mmm	DD 2A n m	IX_H←(mn+1) IX_L←(mn)	–	–	X	–	X	–	–	–	6	20
	LD IY,(mn)	11 111 101 00 101 010 nn nnn nnn mm mmm mmm	FD 2A n m	IY_H←(mn+1) IY_L←(mn)	–	–	X	–	X	–	–	–	6	20
	LD (mn),HL	00 100 010 nn nnn nnn mm mmm mmm	22 n m	(mn+1)←H (mn)←L	–	–	X	–	X	–	–	–	5	16
	LD (mn),t	11 101 101 01 tt0 011 nn nnn nnn mm mmm mmm	ED 43+t×10 n m	(mn+1)←t_H (mn)←t_L	–	–	X	–	X	–	–	–	6	20
	LD (mn),IX	11 011 101 00 100 010 nn nnn nnn mm mmm mmm	DD 22 n m	(mn+1)←IX_H (mn)←IX_L	–	–	X	–	X	–	–	–	6	20
	LD (mn),IY	11 111 101 00 100 010 nn nnn nnn mm mmm mmm	FD 22 n m	(mn+1)←IY_H (mn)←IY_L	–	–	X	–	X	–	–	–	6	20
	LD SP,HL	11 111 001	F9	SP←HL	–	–	X	–	X	–	–	–	1	6
	LD SP,IX	11 011 101 11 111 001	DD F9	SP←IX	–	–	X	–	X	–	–	–	2	10
	LD SP,IY	11 111 101 11 111 001	FD F9	SP←IY	–	–	X	–	X	–	–	–	2	10
	PUSH q	11 qq0 101	C5+q×10	(SP−2)←q_L,(SP−1)←q_H, SP←SP−2	–	–	X	–	X	–	–	–	3	11
	PUSH IX	11 011 101 11 100 101	DD E5	(SP−2)←IX_L,(SP−1)←IX_H SP←SP−2	–	–	X	–	X	–	–	–	4	15
	PUSH IY	11 111 101 11 100 101	FD E5	(SP−2)←IY_L,(SP−1)←IY_H SP←SP−2	–	–	X	–	X	–	–	–	4	15
	POP q	11 qq0 001	C1+q×10	q_H←(SP+1),q_L←(SP), SP←SP+2	–	–	X	–	X	–	–	–	3	10
	POP IX	11 011 101 11 100 001	DD E1	IX_H←(SP+1),IX_L←(SP) SP←SP+2	–	–	X	–	X	–	–	–	4	14
	POP IY	11 111 101 11 100 001	FD E1	IY_H←(SP+1),IY_L←(SP) SP←SP+2	–	–	X	–	X	–	–	–	4	14
*1	EX DE,HL	11 101 011	EB	DE↔HL	–	–	X	–	X	–	–	–	1	4
	EX AF,AF'	00 001 000	08	AF↔AF'	–	–	X	–	X	–	–	–	1	4
	EXX	11 011 001	D9	BC↔BC',DE↔DE',HL↔HL'	–	–	X	–	X	–	–	–	1	4

t	tt
BC	00
DE	01
HL	10
SP	11

q	qq
BC	00
DE	01
HL	10
AF	11

Note : t is any of the register pairs BC, DE, HL, SP.
q is any of the register pairs AF, BC, DE, HL.
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively. (Ex) BC_L = C, AF_H = A.

*1 : EXCHANGE

120489

TMPZ84C00A Instruction Set (3/9)

ITEM/ CLASSI-FICATION	Assembler mnemonic	Object code Binary 76 543 210	Object code Hex	Function	Flag S	Z		H		P/V	N	C	No. OF CYCLES	No. OF STATES	
EXCHANGE	EX (SP),HL	11 100 011	E3	H↔(SP+1),L↔(SP)	–	–	X	–	X	–	–	–	5	19	
	EX (SP),IX	11 011 101	DD	IX_H↔(SP+1)	–	–	X	–	X	–	–	–	6	23	
		11 100 011	E3	IX_L↔(SP)											
	EX (SP),IY	11 111 101	FD	IY_H↔(SP+1)	–	–	X	–	X	–	–	–	6	23	
		11 100 011	E3	IY_L↔(SP)											
BLOCK TRANSFER BLOCK SEARCH	LDI	11 101 101	ED	(DE)←(HL),DE←DE+1	–	–	X	0	X	*M	0	–	4	16	
		10 100 000	A0	HL←HL+1,BC←BC−1											
	LDIR	11 101 101	ED	(DE)←(HL),DE←DE+1	–	–	X	0	X	0	0	–	5	21	↑[BC< >0]
		10 110 000	B0	HL←HL+1,BC←BC−1 Repeat until BC=0									4	16	↑[BC=0]
	LDD	11 101 101	ED	(DE)←(HL),DE←DE−1	–	–	X	0	X	*M	0	–	4	16	
		10 101 000	A8	HL←HL−1,BC←BC−1											
	LDDR	11 101 101	ED	(DE)←(HL),DE←DE−1	–	–	X	0	X	0	0	–	5	21	↑[BC< >0]
		10 111 000	B8	HL←HL−1,BC←BC−1 Repeat until BC=0									4	16	↑[BC=0]
	CPI	11 101 101	ED	A−(HL)	*	*N	X	*	X	*M	1	–	4	16	
		10 100 001	A1	HL←HL+1,BC←BC−1											
	CPIR	11 101 101	ED	A−(HL),HL←HL+1,BC←BC−1	*	*N	X	*	X	*M	1	–	5	21	↑[BC< >0 & A< > (HL)]
		10 110 001	B1	Repeat until A=(HL) or BC=0									4	16	↖[BC=0 or A= (HL)]
	CPD	11 101 101	ED	A−(HL)	*	*N	X	*	X	*M	1	–	4	16	
		10 101 001	A9	HL←HL−1,BC←BC−1											
	CPDR	11 101 101	ED	A−(HL),HL←HL−1,BC←BC−1	*	*N	X	*	X	*M	1	–	5	21	↑[BC< >0 & A< > (HL)]
		10 111 001	B9	Repeat until A=(HL)or BC=0									4	16	↖[BC=0 or A= (HL)]
8-BIT ARITHMETIC AND LOGICAL	ADD A,r	10 000 rrr	80+r	A←A+r	*	*	X	*	X	V	0	*	1	4	
	ADD A,n	11 000 110	C6	A←A+n	*	*	X	*	X	V	0	*	2	7	
		nn nnn nnn	n												
	ADD A,(HL)	10 000 110	86	A←A+(HL)	*	*	X	*	X	V	0	*	2	7	
	ADD A,(IX+d)	11 011 101	DD	A←A+(IX+d)	*	*	X	*	X	V	0	*	5	19	
		10 000 110	86												
		dd ddd ddd	d												
	ADD A,(IY+d)	11 111 101	FD	A←A+(IY+d)	*	*	X	*	X	V	0	*	5	19	
		10 000 110	86												
		dd ddd ddd	d												
	ADC A,r	10 001 rrr	88+r	A←A+r+CY	*	*	X	*	X	V	0	*	1	4	
	ADC A,n	11 001 110	CE	A←A+n+CY	*	*	X	*	X	V	0	*	2	7	
		nn nnn nnn	n												
	ADC A,(HL)	10 001 110	8E	A←A+(HL)+CY	*	*	X	*	X	V	0	*	2	7	
	ADC A,(IX+d)	11 011 101	DD	A←A+(IX+d)+CY	*	*	X	*	X	V	0	*	5	19	
		10 001 110	8E												
		dd ddd ddd	d												
	ADC A,(IY+d)	11 111 101	FD	A←A+(IY+d)+CY	*	*	X	*	X	V	0	*	5	19	
		10 001 110	8E												
		dd ddd ddd	d												
	SUB r	10 010 rrr	90+r	A←A−r	*	*	X	*	X	V	1	*	1	4	
	SUB n	11 010 110	D6	A←A−n	*	*	X	*	X	V	1	*	2	7	
		nn nnn nnn	n												
	SUB (HL)	10 010 110	96	A←A−(HL)	*	*	X	*	X	V	1	*	2	7	
	SUB (IX+d)	11 011 101	DD	A←A−(IX+d)	*	*	X	*	X	V	1	*	5	19	
		10 010 110	96												
		dd ddd ddd	d												
	SUB (IY+d)	11 111 101	FD	A←A−(IY+d)	*	*	X	*	X	V	1	*	5	19	
		10 010 110	96												
		dd ddd ddd	d												

r	rrr
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Note : *M P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1.
*N Z flag is 1 if A = (HL), otherwise Z = 0.
[ ] indicates the total condition of the number of cycles and states indicated by arrow.
r means any of the registers A, B, C, D, E, H, L.

120489

TMPZ84C00A Instruction Set (4/9)

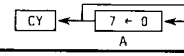
ITEM/ CLASSIFICATION	Assembler mnemonic	Object code Binary (76 543 210)	Object code Hex	Function	Flag S	Z		H		P/V	N	C	No. OF CYCLES	No. OF STATES
8-BIT ARITHMETIC AND LOGICAL	SBC A,r	10 011 rrr	98+r	A←A−r−CY	*	*	X	*	X	V	1	*	1	4
	SBC A,n	11 011 110 nn nnn nnn	DE n	A←A−n−CY	*	*	X	*	X	V	1	*	2	7
	SBC A,(HL)	10 011 110	9E	A←A−(HL)−CY	*	*	X	*	X	V	1	*	2	7
	SBC A,(IX+d)	11 011 101 10 011 110 dd ddd ddd	DD 9E d	A←A−(IX+d)−CY	*	*	X	*	X	V	1	*	5	19
	SBC A,(IY+d)	11 111 101 10 011 110 dd ddd ddd	FD 9E d	A←A−(IY+d)−CY	*	*	X	*	X	V	1	*	5	19
	AND r	10 100 rrr	A0+r	A←A∧r	*	*	X	1	X	P	0	0	1	4
	AND n	11 100 110 nn nnn nnn	E6 n	A←A∧n	*	*	X	1	X	P	0	0	2	7
	AND (HL)	10 100 110	A6	A←A∧(HL)	*	*	X	1	X	P	0	0	2	7
	AND (IX+d)	11 011 101 10 100 110 dd ddd ddd	DD A6 d	A←A∧(IX+d)	*	*	X	1	X	P	0	0	5	19
	AND (IY+d)	11 111 101 10 100 110 dd ddd ddd	FD A6 d	A←A∧(IY+d)	*	*	X	1	X	P	0	0	5	19
	OR r	10 110 rrr	B0+r	A←A∨r	*	*	X	0	X	P	0	0	1	4
	OR n	11 110 110 nn nnn nnn	F6 n	A←A∨n	*	*	X	0	X	P	0	0	2	7
	OR (HL)	10 110 110	B6	A←A∨(HL)	*	*	X	0	X	P	0	0	2	7
	OR (IX+d)	11 011 101 10 110 110 dd ddd ddd	DD B6 d	A←A∨(IX+d)	*	*	X	0	X	P	0	0	5	19
	OR (IY+d)	11 111 101 10 110 110 dd ddd ddd	FD B6 d	A←A∨(IY+d)	*	*	X	0	X	P	0	0	5	19
	XOR r	10 101 rrr	A8+r	A←A∀r	*	*	X	0	X	P	0	0	1	4
	XOR n	11 101 110 nn nnn nnn	EE n	A←A∀n	*	*	X	0	X	P	0	0	2	7
	XOR (HL)	10 101 110	AE	A←A∀(HL)	*	*	X	0	X	P	0	0	2	7
	XOR (IX+d)	11 011 101 10 101 110 dd ddd ddd	DD AE d	A←A∀(IX+d)	*	*	X	0	X	P	0	0	5	19
	XOR (IY+d)	11 111 101 10 101 110 dd ddd ddd	FD AE d	A←A∀(IY+d)	*	*	X	0	X	P	0	0	5	19
	CP r	10 111 rrr	B8+r	A−r	*	*	X	*	X	V	1	*	1	4
	CP n	11 111 110 nn nnn nnn	FE n	A−n	*	*	X	*	X	V	1	*	2	7
	CP (HL)	10 111 110	BE	A−(HL)	*	*	X	*	X	V	1	*	2	7
	CP (IX+d)	11 011 101 10 111 110 dd ddd ddd	DD BE d	A−(IX+d)	*	*	X	*	X	V	1	*	5	19
	CP (IY+d)	11 111 101 10 111 110 dd ddd ddd	FD BE d	A−(IY+d)	*	*	X	*	X	V	1	*	5	19
	INC r	00 rrr 100	04+r×8	r←r+1	*	*	X	*	X	V	0	−	1	4
	INC (HL)	00 110 100	34	(HL)←(HL)+1	*	*	X	*	X	V	0	−	3	11
	INC (IX+d)	11 011 101 00 110 100 dd ddd ddd	DD 34 d	(IX+d)←(IX+d)+1	*	*	X	*	X	V	0	−	6	23

r	rrr
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Note : r means any of the registers A, B, C, D, E, H, L.

120489

TMPZ84C00A Instruction Set (5/9)

ITEM/CLASSIFICATION	Assembler mnemonic	Object code Binary (76 543 210)	Object code Hex	Function	Flag S	Z		H		P/V	N	C	No. OF CYCLES	No. OF STATES
8-BIT ARITHMETIC AND LOGICAL	INC (IY+d)	11 111 101 00 110 100 dd ddd ddd	FD 34 d	(IY+d)←(IY+d)+1	*	*	X	*	X	V	0	–	6	23
	DEC r	00 rrr 101	05+r×8	r←r−1	*	*	X	*	X	V	1	–	1	4
	DEC (HL)	00 110 101	35	(HL)←(HL)−1	*	*	X	*	X	V	1	–	3	11
	DEC (IX+d)	11 011 101 00 110 101 dd ddd ddd	DD 35 d	(IX+d)←(IX+d)−1	*	*	X	*	X	V	1	–	6	23
	DEC (IY+d)	11 111 101 00 110 101 dd ddd ddd	FD 35 d	(IY+d)←(IY+d)−1	*	*	X	*	X	V	1	–	6	23
GENELAL-PURPOSE ARITHMETIC AND MPU CONTROL	DAA	00 100 111	27	Decimal adjust accumulator	*	*	X	*	X	P	–	*	1	4
	CPL	00 101 111	2F	$A \leftarrow \overline{A}$	–	–	X	1	X	–	1	–	1	4
	NEG	11 101 101 01 000 100	ED 44	A←0−A	*	*	X	*	X	V	1	*	2	8
	CCF	00 111 111	3F	$CY \leftarrow \overline{CY}$	–	–	X	X	X	–	0	*	1	4
	SCF	00 110 111	37	CY←1	–	–	X	0	X	–	0	1	1	4
	NOP	00 000 000	00	no operation	–	–	X	–	X	–	–	–	1	4
	HALT	01 110 110	76	MPU Halted	–	–	X	–	X	–	–	–	1	4
	DI	11 110 011	F3	IFF←0	–	–	X	–	X	–	–	–	1	4
	EI	11 111 011	FB	IFF←1	–	–	X	–	X	–	–	–	1	4
	IM 0	11 101 101 01 000 110	ED 46	Set interrupt mode 0	–	–	X	–	X	–	–	–	2	8
	IM 1	11 101 101 01 010 110	ED 56	Set interrupt mode 1	–	–	X	–	X	–	–	–	2	8
	IM 2	11 101 101 01 011 110	ED 5E	Set interrupt mode 2	–	–	X	–	X	–	–	–	2	8
16-BIT ARITHMETIC	ADD HL,t	00 tt1 001	09+t×10	HL←HL+t	–	–	X	X	X	–	0	*	3	11
	ADC HL,t	11 101 101 01 tt1 010	ED 4A+t×10	HL←HL+t+CY	*	*	X	X	X	V	0	*	4	15
	SBC HL,t	11 101 101 01 tt0 010	ED 42+t×10	HL←HL−t−CY	*	*	X	X	X	V	1	*	4	15
	ADD IX,p	11 011 101 00 pp1 001	DD 09+p×10	IX←IX+p	–	–	X	X	X	–	0	*	4	15
	ADD IY,s	11 111 101 00 ss1 001	FD 09+s×10	IY←IY+s	–	–	X	X	X	–	0	*	4	15
	INC t	00 tt0 011	03+t×10	t←t+1	–	–	X	–	X	–	–	–	1	6
	INC IX	11 011 101 00 100 011	DD 23	IX←IX+1	–	–	X	–	X	–	–	–	2	10
	INC IY	11 111 101 00 100 011	FD 23	IY←IY+1	–	–	X	–	X	–	–	–	2	10
	DEC t	00 tt1 011	0B+t×10	t←t−1	–	–	X	–	X	–	–	–	1	6
	DEC IX	11 011 101 00 101 011	DD 2B	IX←IX−1	–	–	X	–	X	–	–	–	2	10
	DEC IY	11 111 101 00 101 011	FD 2B	IY←IY−1	–	–	X	–	X	–	–	–	2	10
ROTATE	RLCA	00 000 111	07		–	–	X	0	X	–	0	*	1	4

r	rrr
B	000
C	001
D	010
E	011
H	100
L	101
A	111

t	tt
BC	00
DE	01
HL	10
SP	11

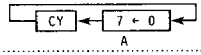
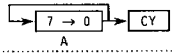
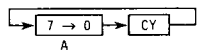
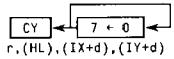
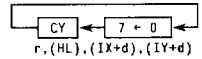
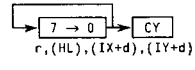
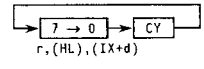
p	pp
BC	00
DE	01
IX	10
SP	11

s	ss
BC	00
DE	01
IY	10
SP	11

Note : ss is any of the register pairs BC, DE, HL, SP. PP is any of the register pairs BC, DE, IX, SP.
rr is any of the register pairs BC, DE, IY, SP.

120489

TMPZ84C00A Instruction Set (6/9)

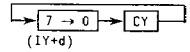
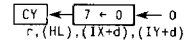
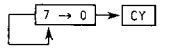
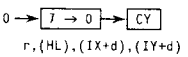
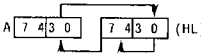
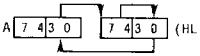
ITEM/ CLASSI-FICA-TION	Assembler mnemonic	Object code Binary 76 543 210	Object code Hex	Function	Flag S	Z		H		P/V	N	C	No. OF CY-CLES	No. OF STA-TES
ROTATE SHIFT	RLA	00 010 111	17		–	–	X	0	X	–	0	*	1	4
	RRCA	00 001 111	0F		–	–	X	0	X	–	0	*	1	4
	RRA	00 011 111	1F		–	–	X	0	X	–	0	*	1	4
	RLC r	11 001 011 00 000 rrr	CB 00+r		*	*	X	0	X	P	0	*	2	8
	RLC (HL)	11 001 011 00 000 110	CB 06		*	*	X	0	X	P	0	*	4	15
	RLC (IX+d)	11 011 101 11 001 011 dd ddd ddd 00 000 110	DD CB d 06		*	*	X	0	X	P	0	*	6	23
	RLC (IY+d)	11 111 101 11 001 011 dd ddd ddd 00 000 110	FD CB d 06		*	*	X	0	X	P	0	*	6	23
	RL r	11 001 011 00 010 rrr	CB 10+r		*	*	X	0	X	P	0	*	2	8
	RL (HL)	11 001 011 00 010 110	CB 16		*	*	X	0	X	P	0	*	4	15
	RL (IX+d)	11 011 101 11 001 011 dd ddd ddd 00 010 110	DD CB d 16		*	*	X	0	X	P	0	*	6	23
	RL (IY+d)	11 111 101 11 001 011 dd ddd ddd 00 010 110	FD CB d 16		*	*	X	0	X	P	0	*	6	23
	RRC r	11 001 011 00 001 rrr	CB 08+r		*	*	X	0	X	P	0	*	2	8
	RRC (HL)	11 001 011 00 001 110	CB 0E		*	*	X	0	X	P	0	*	4	15
	RRC (IX+d)	11 011 101 11 001 011 dd ddd ddd 00 001 110	DD CB d 0E		*	*	X	0	X	P	0	*	6	23
	RRC (IY+d)	11 111 101 11 001 011 dd ddd ddd 00 001 110	FD CB d 0E		*	*	X	0	X	P	0	*	6	23
	RR r	11 001 011 00 011 rrr	CB 18+r		*	*	X	0	X	P	0	*	2	8
	RR (HL)	11 001 011 00 011 110	CB 1E		*	*	X	0	X	P	0	*	4	15
	RR (IX+d)	11 011 101 11 001 011 dd ddd ddd 00 011 110	DD CB d 1E		*	*	X	0	X	P	0	*	6	23

r	rrr
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Note : r means any of the registers A, B, C, D, E, H, L.

120489

TMPZ84C00A Instruction Set (7/9)

ITEM/ CLASSI-FICATION	Assembler mnemonic	Object code Binary	Object code Hex	Function	Flag S	Z		H		P/V	N	C	No. OF CY-CLES	No. OF STA-TES
ROTATE SHIFT	RR (IY+d)	11 111 101 11 001 011 dd ddd ddd 00 011 110	FD CB d 1E	 (IY+d)	*	*	X	0	X	P	0	*	6	23
	SLA r	11 001 011 00 100 rrr	CB 20+r		*	*	X	0	X	P	0	*	2	8
	SLA (HL)	11 001 011 00 100 110	CB 26		*	*	X	0	X	P	0	*	4	15
	SLA (IX+d)	11 011 101 11 001 011 dd ddd ddd 00 100 110	DD CB d 26	 r,(HL),(IX+d),(IY+d)	*	*	X	0	X	P	0	*	6	23
	SLA (IY+d)	11 111 101 11 001 011 dd ddd ddd 00 100 110	FD CB d 26		*	*	X	0	X	P	0	*	6	23
	SRA r	11 001 011 00 101 rrr	CB 28+r		*	*	X	0	X	P	0	*	2	8
	SRA (HL)	11 001 011 00 101 110	CB 2E		*	*	X	0	X	P	0	*	4	15
	SRA (IX+d)	11 011 101 11 001 011 dd ddd ddd 00 101 110	DD CB d 2E	 r,(HL),(IX+d),(IY+d)	*	*	X	0	X	P	0	*	6	23
	SRA (IY+d)	11 111 101 11 001 011 dd ddd ddd 00 101 110	FD CB d 2E		*	*	X	0	X	P	0	*	6	23
	SRL r	11 001 011 00 111 rrr	CB 38+r		*	*	X	0	X	P	0	*	2	8
	SRL (HL)	11 001 011 00 111 110	CB 3E		*	*	X	0	X	P	0	*	4	15
	SRL (IX+d)	11 011 101 11 001 011 dd ddd ddd 00 111 110	DD CB d 3E	 r,(HL),(IX+d),(IY+d)	*	*	X	0	X	P	0	*	6	23
	SRL (IY+d)	11 111 101 11 001 011 dd ddd ddd 00 111 110	FD CB d 3E		*	*	X	0	X	P	0	*	6	23
	RLD	11 101 101 01 101 111	ED 6F		*	*	X	0	X	P	0	-	5	18
	RRD	11 101 101 01 100 111	ED 67		*	*	X	0	X	P	0	-	5	18
BIT SET RESET AND TEST	BIT b,r	11 001 011 01 bbb rrr	CB 40+b×8+r	$Z \leftarrow \overline{r_b}$	X	*	X	1	X	X	0	-	2	8
	BIT b,(HL)	11 001 011 01 bbb 110	CB 46+b×8	$Z \leftarrow \overline{(HL)_b}$	X	*	X	1	X	X	0	-	3	12

*1

r	rrr
B	000
C	001
D	010
E	011
H	100
L	101
A	111

b	bbb
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Note : *1 : Rotate digit left and right between the accmulator and location (HL).
The content of the upper half of the accumulator is unaffected.
The notation $(HL)_b$ indicates bit $_b$ (0 to 7) within the contents of the HL register pair.
The notation r_b indicates bit $_b$ (0 to 7) within the r register.

120489

TMPZ84C00A Instruction Set (8/9)

ITEM/ CLASSI -FICA- TION	Assembler mnemonic	Object code Binary 76 543 210	Object code Hex	Function	Flag S	Z		H		P/V	N	C	No. OF CY- CLES	No. OF STA- TES
BIT SET RESET AND TEST	BIT b,(IX+d)	11 011 101	DD		X	*	X	1	X	X	0	–	5	20
		11 001 011	CB	$Z \leftarrow \overline{(IX+d)_b}$										
		dd ddd ddd	d											
		01 bbb 110	46+b×8											
	BIT b,(IY+d)	11 111 101	FD		X	*	X	1	X	X	0	–	5	20
		11 001 011	CB	$Z \leftarrow \overline{(IY+d)_b}$										
		dd ddd ddd	d											
		01 bbb 110	46+b×8											
	SET b,r	11 001 011	CB	$r_b \leftarrow 1$	–	–	X	–	X	–	–	–	2	8
		11 bbb rrr	C0+b×8+r											
	SET b,(HL)	11 001 011	CB	$(HL)_b \leftarrow 1$	–	–	X	–	X	–	–	–	4	15
		11 bbb 110	C6+b×8											
	SET b,(IX+d)	11 011 101	DD	$(IX+d)_b \leftarrow 1$	–	–	X	–	X	–	–	–	6	23
		11 001 011	CB											
		dd ddd ddd	d											
		11 bbb 110	C6+b×8											
	SET b,(IY+d)	11 111 101	FD	$(IY+d)_b \leftarrow 1$	–	–	X	–	X	–	–	–	6	23
		11 001 011	CB											
		dd ddd ddd	d											
		11 bbb 110	C6+b×8											
	RES b,r	11 001 011	CB	$r_b \leftarrow 0$	–	–	X	–	X	–	–	–	2	8
		10 bbb rrr	80+b×8+r											
	RES b,(HL)	11 001 011	CB	$(HL)_b \leftarrow 0$	–	–	X	–	X	–	–	–	4	15
		10 bbb 110	86+b×8											
	RES b,(IX+d)	11 011 101	DD	$(IX+d)_b \leftarrow 0$	–	–	X	–	X	–	–	–	6	23
		11 001 011	CB											
		dd ddd ddd	d											
		10 bbb 110	86+b×8											
	RES b,(IY+d)	11 111 101	FD	$(IY+d)_b \leftarrow 0$	–	–	X	–	X	–	–	–	6	23
		11 001 011	CB											
		dd ddd ddd	d											
		10 bbb 110	86+b×8											
JUMP	JP mn	11 000 011	C3	PC←mn	–	–	X	–	X	–	–	–	3	10
		nn nnn nnn	n											
		mm mmm mmm	m											
	JP c,mn	11 ccc 010	C2+c×8	PC←mn	–	–	X	–	X	–	–	–	3	10
		nn nnn nnn	n	(Only when condition is met)										
		mm mmm mmm	m											
	JR $+e	00 011 000	18	PC←$+e	–	–	X	–	X	–	–	–	3	12
		aa aaa aaa	a											
	JR C,$+e	00 111 000	38	If C=0, continue	–	–	X	–	X	–	–	–	2	7
		aa aaa aaa	a	If C=1, PC←$+e	–	–	X	–	X	–	–	–	3	12
	JR NC,$+e	00 110 000	30	If C=0, PC←$+e	–	–	X	–	X	–	–	–	3	12
		aa aaa aaa	a	If C=1, continue	–	–	X	–	X	–	–	–	2	7
	JR Z,$+e	00 101 000	28	If Z=0, continue	–	–	X	–	X	–	–	–	2	7
		aa aaa aaa	a	If Z=1, PC←$+e	–	–	X	–	X	–	–	–	3	12
	JR NZ,$+e	00 100 000	20	If Z=0, PC←$+e	–	–	X	–	X	–	–	–	3	12
		aa aaa aaa	a	If Z=1, continue	–	–	X	–	X	–	–	–	2	7
	DJNZ $+e	00 010 000	10	B←B−1, If B=0, continue	–	–	X	–	X	–	–	–	2	8
		aa aaa aaa	a	B←B−1, If B<>0, continue	–	–	X	–	X	–	–	–	3	13
	JP (HL)	11 101 001	E9	PC←HL	–	–	X	–	X	–	–	–	1	4

r	rrr
B	000
C	001
D	010
E	011
H	100
L	101
A	111

b	bbb
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

e represents the extension in the relative addressing mode, a = e − 2. e is a signed two's complement number in the range of −126 ≤ e ≤ 129

Note :
- a = e − 2 in the Opcode provides an effective address of PC + e as PC is incremented by 2 before the addition of e.
- $ indicates the reference to the location counter value of the current segment.
- The notation $(HL)_b$, $(IX + d)_b$ indicates bit b (0 to 7) within the contents of the register pair.
- The notation r_b indicates bit b (0 to 7) within the r register.
- a = e-2 in the op-code provides effective address of PC + e as PC is incremented by 2 prior to the addition of e.

c	ccc	Condition
NZ	000	Non-Zero
Z	001	Zero
NC	010	No-carry
C	011	Carry
PO	100	Odd Parity
PE	101	Even Parity
P	110	Sign Positive
M	111	Sign Negative

270489

TMPZ84C00A Instruction Set (9/9)

ITEM/ CLASSI-FICATION	Assembler mnemonic	Object code Binary (76 543 210)	Object code Hex	Function	Flag S	Z		H		P/V	N	C	No. OF CYCLES	No. OF STATES
JUMP	JP (IX)	11 011 101 11 101 001	DD E9	PC←(IX)	–	–	X	–	X	–	–	–	2	8
	JP (IY)	11 111 101 11 101 001	FD E9	PC←(IY)	–	–	X	–	X	–	–	–	2	8
CALL AND RETURN	CALL mn	11 001 101 nn nnn nnn mm mmm mmm	CD n m	(SP-1)←PC_H, (SP-2)←PC_L PC←mn SP←SP-2	–	–	X	–	X	–	–	–	5	17
	CALL c,mn	11 ccc 100 nn nnn nnn mm mmm mmm	C4+cX8 n m	If condition c is met, same as CALL mn. If condition c is not met, continue	– –	– –	X X	– –	X X	– –	– –	– –	5 3	17 10
	RET	11 001 001	C9	PC_L←(SP), PC_H←(SP+1) SP←SP+2	–	–	X	–	X	–	–	–	3	10
	RET c	11 ccc 000	C0+cx8	If condition c is met, same as RET. If condition c is not met, continue	– –	– –	X X	– –	X X	– –	– –	– –	3 1	11 5
	RETI	11 101 101 01 001 101	ED 4D	Return from interrupt Processing routine	–	–	X	–	X	–	–	–	4	14
	RETN	11 101 101 01 000 101	ED 45	Return from non-maskable interrupt Processing routine	–	–	X	–	X	–	–	–	4	14
	RST j	11 kkk 111	C7+kx8	(SP-1)←PC_H, (SP-2)←PC_L PC_H←0, PC_L←j, SP←SP-2	–	–	X	–	X	–	–	–	3	11
INPUT AND OUTPUT	IN A,(n)	11 011 011 nn nnn nnn	DB n	A←(n) n→A0~A7, A→A8~A15	–	–	X	–	X	–	–	–	3	11
	IN r,(C)	11 101 101 01 rrr 000	ED 40+rx8	r←(C) If r=110, only the flags will be affected.	*	*	X	*	X	P	0	–	3	12
	INI	11 101 101 10 100 010	ED A2	(HL)←(C), B←B-1, HL←HL+1	X	*M	X	X	X	X	1	X	4	16
	INIR	11 101 101 10 110 010	ED B2	(HL)←(C), B←B-1, HL←HL+1 Repeat until B=0	X	1	X	X	X	X	1	X	5 4	21 16
	IND	11 101 101 10 101 010	ED AA	(HL)←(C), B←B-1, HL←HL-1	X	*M	X	X	X	X	1	X	4	16
	INDR	11 101 101 10 111 010	ED BA	(HL)←(C), B←B-1, HL←HL-1 Repeat until B=0	X	1	X	X	X	X	1	X	5 4	21 16
	OUT (n),A	11 010 011 nn nnn nnn	D3 n	(n)←A n→A0~A7, A→A8~A15	–	–	X	–	X	–	–	–	3	11
	OUT (C),r	11 101 101 01 rrr 001	ED 41+rX8	(C)←r	–	–	X	–	X	–	–	–	3	12
	OUTI	11 101 101 10 100 011	ED A3	(C)←(HL), B←B-1, HL←HL+1	X	*M	X	X	X	X	1	X	4	16
	OTIR	11 101 101 10 110 011	ED B3	(C)←(HL), B←B-1, HL←HL+1 Repeat until B=0	X	1	X	X	X	X	1	X	5 4	21 16
	OUTD	11 101 101 10 101 011	ED AB	(C)←(HL), B←B-1, HL←HL-1	X	*M	X	X	X	X	1	X	4	16
	OTDR	11 101 101 10 111 010	ED BB	(C)←(HL), B←B-1, HL←HL-1 Repeat until B=0	X	1	X	X	X	X	1	X	5 4	21 16

j	kkk
00H	000
08H	001
10H	010
18H	011
20H	100
28H	101
30H	110
38H	111

r	rrr
B	000
C	001
D	010
E	011
H	100
L	101
A	111

←[B<>0]
←[B=0] *1

←[B<>0]
←[B=0]

←B[<>0]
←[B=0] *1

←[B<>0]
←[B=0]

Note :
- *M If the result of B – 1 is zero, the Z flag is set, otherwise it is reset.
- A0 through A15 indicate the address bus.
- [ ] indicates the total condition of the number of cycles and states indicated by arrow.

*1 C→A0~A7
B→A8~A15

c	ccc	Condition
NZ	000	Non-Zero
Z	001	Zero
NC	010	No-Carry
C	011	Carry
PO	100	Odd Parity
PE	101	Even Parity
P	110	Sign Positive
M	111	Sign negative

120489

3.5 USAGE

Basic TMPZ84C00A configurations using memory and peripheral LSIs are described below.

3.5.1 Memory Address Assignment

When the memory is being accessed, the MPU outputs address and control signals. These signals are used as the memory chip enable signals.

The MPU uses 16-bit address signals to specify the addresses for 64K (0-FFFF). With systems having only one memory, memory addresses can be specified with these signals alone. When there are several memories, however, the memories must be arranged so that access is possible using 64K of space. Normally, several address buses are decoded to create this arrangement, several address buses are developed for use as one memory chip enable signal for all memories.

Example: The addresses for an 8K×8-bit ROM and 8K×8-bit RAM are arranged as shown in Figure 3.21. Figure 3.22 shows am example using the $\overline{MREQ}$ signal, $\overline{RD}$ signal and address signal A13 as the chip enable signals.

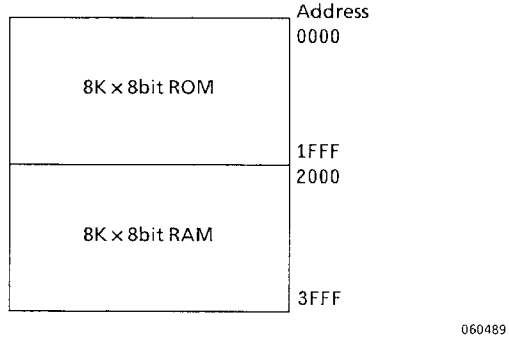


Figure 3.21 Address Assignment

3.5.2 Connection with TLCS-Z80 family peripheral LSI

TMPZ84C00A can connect with peripheral LSI directly. A simple connecting example of the TMPZ84C00A with peripheral LSI is shown in Figure 3.23.

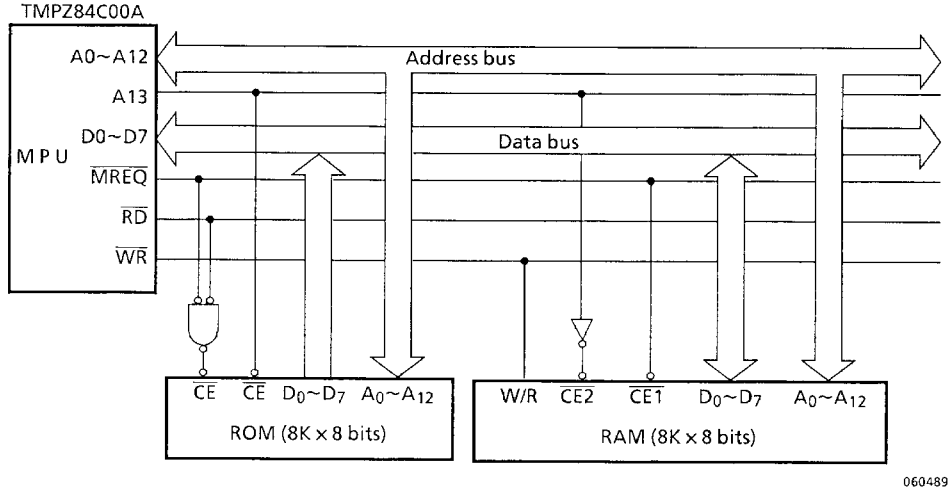
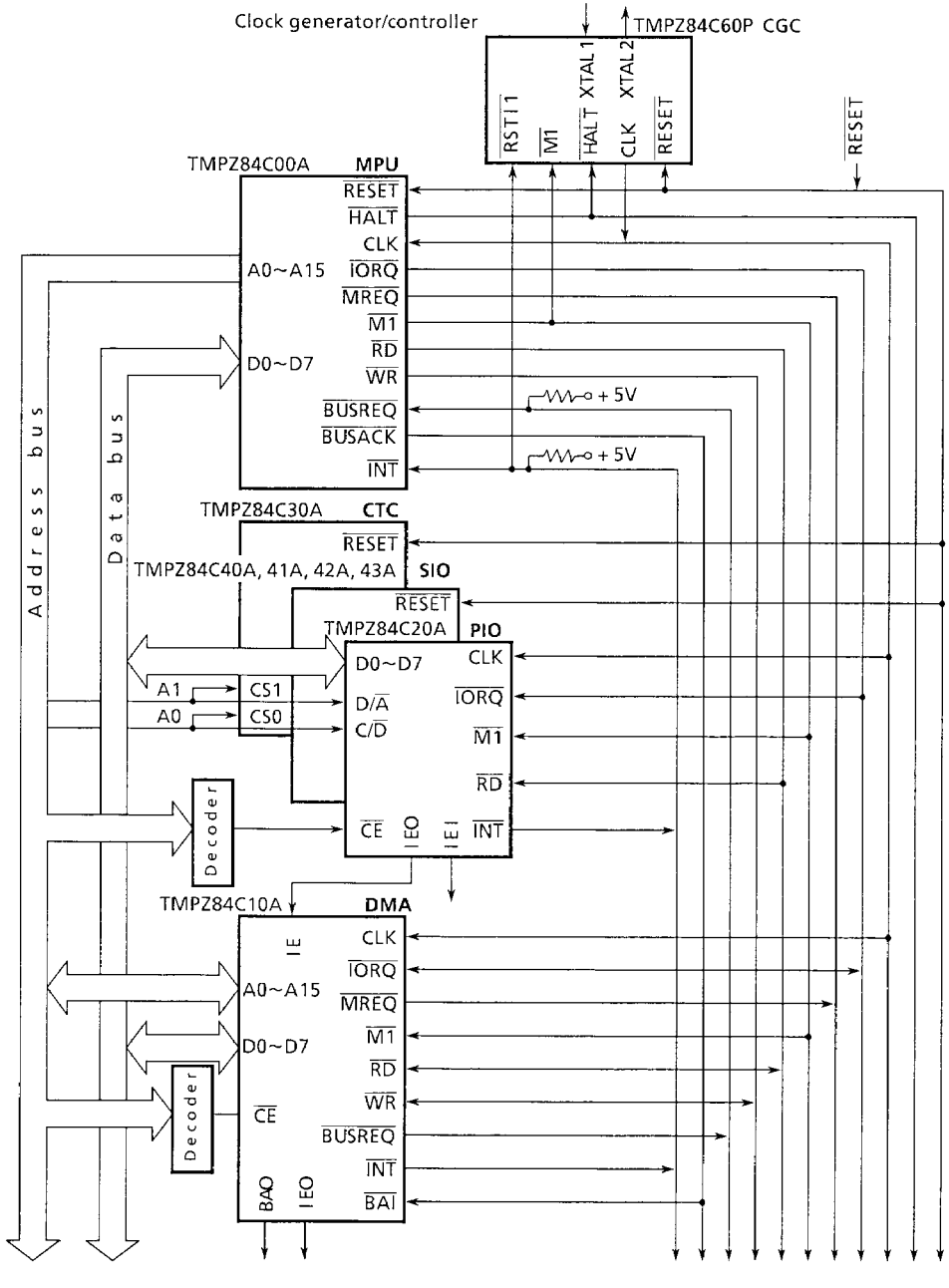


Figure 3.22 Example Connection with Memories



060489

Figure 3.23 Example Connection with TLCS-Z80 family peripheral LSIs

4. ELECTRICAL CHARACTERISTICS

4.1 ABSOLUTE MAXIMUM RATINGS

SYMBOL	ITEM	RATING	UNIT
V_{CC}	Supply Voltage	−0.5~+7	V
V_{IN}	Input Voltage	−0.5~VCC+0.5	V
PD	Power Dissipation (TA = 85°C)	250	mW
T_{SOLDER}	Soldering Temperature (10sec)	260	°C
TSTG	Storage Temperature	−65~150	°C
TOPR	Operating Temprature	−40~85	°C

060489

4.2 DC ELECTRICLAL CHARACTERISTICS

DC Characteristics (1/2)
T_{OPR} = −40°C~85°C, V_{CC} = 5V ± 10%, VSS = 0V

SYMBOL	ITEM	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
VILC	Low Level Clock input Voltage		−0.3	—	0.6	V
VIHC	High Level Clock input Voltage		V_{CC} − 0.6	—	V_{CC} + 0.3	V
VIL	Input Low Voltage (except CLK)		−0.5	—	0.8	V
VIH	Input High Voltage (except CLK)		2.2	—	V_{CC}	V
VOL	Output Low Voltage	IOL = 2.0mA	—	—	0.4	V
VOH1	Output High Voltage (I)	IOH = -1.6mA	2.4	—	—	V
VOH2	Output High Voltage (II)	IOH = -250μA	V_{CC} − 0.8	—	—	V
ILI	Input Leak Current	$V_{SS} \leqq V_{IN} \leqq V_{CC}$	—	—	±10	μA
ILO	3 state Output current in Floating	V_{SS} + 0.4 ≦ V_{OUT} ≦ V_{CC}	—	—	±10	μA

060489

DC Characteristics (2/2)

SYMBOL	ITEM	TEST CONDITION		MIN.	TYP.	MAX.	UNIT
$I_{CC}1$	Supply Current (Operating)	V_{CC} = 5V fCLK = (Note1) VIHC = VIH = V_{CC} − 0.2V VILC = VIL = 0.2V	AP-6 /AM-6 /AT-6	—	15	22	mA
			AP-8 /AM-8 /AT-8	—	20	25	
$I_{CC}2$ (Note2)	Supply Current (Stand by)	V_{CC} = 5V CLK = (Note2) VIHC = VIH = V_{CC}-0.2V VILC = VIL = 0.2V		—	0.5	10	μA

Note 1 $f_{CLK} = 1/T_{CC}$ (MIN.)
Note 2 At T4 "LOW" state after the halt instruciton fetch cycle.

060489

4.3 AC ELECTRICAL CHARACTERISTICS (1/3)

$T_{OPR} = -40°C \sim 85°C$, $V_{CC} = 5V \pm 10\%$, $VSS = 0V$

NO.	SYMBOL	ITEM	AP-6/AM-6 /AT-6 (6MHz)		AP-8/AM-8 /AT-8 (8MHz)		UNIT
			MIN.	MAX.	MIN.	MAX.	
1	TcC	Clock Cycle Time	165	DC	125	DC	ns
2	TwCh	Clock Pulse Width (High)	65	DC	55	DC	ns
3	TwC1	Clock Pulse Width (Low)	65	DC	55	DC	ns
4	TfC	Clock Fall Time	—	20	—	10	ns
5	TrC	Clock Rise Time	—	20	—	10	ns
6	TdCr (A)	Clock ↑ to Address Valid Delay	—	90	—	80	ns
7	TdA (MREQf)	Address Valid to $\overline{MREQ}$ ↓ Delay	35	—	20	—	ns
8	TdCf (MREQf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay	—	70	—	60	ns
9	TdCr (MREQr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay	—	70	—	60	ns
10	TwMREQh	$\overline{MREQ}$ pulse Width (High)	65	—	45	—	ns
11	TwMREQ1	$\overline{MREQ}$ pulse Width (Low)	135	—	100	—	ns
12	TdCf (MREQr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay	—	70	—	60	ns
13	TdCf (RDf)	Clock ↓ to $\overline{RD}$ ↓ Delay	—	80	—	70	ns
14	TdCr (RDr)	Clock ↑ to $\overline{RD}$ ↑ Delay	—	70	—	60	ns
15	TsD (Cr)	Data Setup Time to Clock ↑	30	—	30	—	ns
16	ThD (RDr)	Data Hold Time to $\overline{RD}$ ↑	0	—	0	—	ns
17	TsWAIT (Cf)	$\overline{WAIT}$ Setup Time to Clock ↓	60	—	50	—	ns

060489

AC Electrical Characteristics (2/3)

NO.	SYMBOL	ITEM	AP-6/AM-6 /AT-6 (6MHz)		AP-8/AM-8 /AT-8 (8MHz)		UNIT
			MIN	MAX.	MIN	MAX.	
18 *	ThWAIT (Cf)	$\overline{WAIT}$ Hold Time after Clock ↓	10	—	10	—	ns
19	TdCr (M1f)	Clock ↑ to $\overline{M1}$ ↓ Delay	—	80	—	70	ns
20	TdCr (M1r)	Clock ↑ to $\overline{M1}$ ↑ Delay	—	80	—	70	ns
21	TdCr (RFSHf)	Clock ↑ to $\overline{RFSH}$ ↓ Delay	—	110	—	95	ns
22	TdCr (RFSHr)	Clock ↑ to $\overline{RFSH}$ ↑ Delay	—	100	—	85	ns
23	TdCf (RDr)	Clock ↓ to $\overline{RD}$ ↑ Delay	—	70	—	60	ns
24	TdCr (RDf)	Clock ↑ to $\overline{RD}$ ↓ Delay	—	70	—	60	ns
25	TsD (Cf)	Data Setup to Clock ↓ during M2, M3, M4 or M5 Cycles	40	—	30	—	ns
26	TdA (IORQf)	Address Stable prior to $\overline{IORQ}$ ↓	110	—	75	—	ns
27	TdCr (IORQf)	Clock ↑ to $\overline{IORQ}$ ↓ Delay	—	65	—	55	ns
28	TdCf (IORQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay	—	70	—	60	ns
29	TdD (WRf)	Data Stable Prior to $\overline{WR}$ ↓	25	—	5	—	ns
30	TdCf (WRf)	Clock ↓ to $\overline{WR}$ ↓ Delay	—	70	—	60	ns
31	TwWR	$\overline{WR}$ Pulse Width	135	—	100	—	ns
32	TdCf (WRr)	Clock ↓ to $\overline{WR}$ ↑ Delay	—	70	—	60	ns
33	TdD (WRf)	Data Stable Prior to $\overline{WR}$ ↓	-55	—	-55	—	ns
34	TdCr (WRf)	Clock ↑ to $\overline{WR}$ ↓ Delay	—	60	—	55	ns
35	TdWRr (D)	Data Stable from $\overline{WR}$ ↑	30	—	15	—	ns
36	TdCf (HALT)	Clock ↓ to $\overline{HALT}$ ↑ or ↓	—	260	—	225	ns
37	TwNMI	$\overline{NMI}$ Pulse Width	70	—	60	—	ns
38	TsBUSREQ (Cr)	$\overline{BUSREQ}$ Setup Time to Clock ↑	50	—	40	—	ns
39 *	ThBUSREQ(Cr)	$\overline{BUSREQ}$ Hold Time after Clock ↑	10	—	10	—	ns
40	TdCr(BUSACKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay	—	90	—	80	ns
41	TdCf(BUSACKr)	Clockt ↓ to $\overline{BUSACK}$ ↑ Delay	—	90	—	80	ns
42	TdCr(Dz)	Clock ↑ to Data Float Delay	—	80	—	70	ns
43	TdCr(CTz)	Clock ↑ to Control Out-puts Float Delay($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$)	—	70	—	60	ns
44	TdCr(Az)	Clock ↑ to Address Float Delay	—	80	—	70	ns

060489

AC Electrical Characteristics (3/3)

NO.	SYMBOL	ITEM	AP-6/AM-6 /AT-6 (6MHz)		AP-8/AM-8 /AT-8 (8MHz)		UNIT
			MIN.	MAX.	MIN.	MAX.	
45	TdCr(A)	$\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ to Address Hold Time	35	—	20	—	ns
46	TsRESET(Cr)	$\overline{RESET}$ to Clock ↑ setup Time	60	—	45	—	ns
47 *	ThRESET(Cr)	$\overline{RESET}$ to Clock ↑ Hold Time	10	—	10	—	ns
48	TsINTf(Cr)	$\overline{INT}$ to Clock ↑ Setup Time	70	—	55	—	ns
49 *	TsINTr(Cr)	$\overline{INT}$ to Clock ↑ Hold Time	10	—	10	—	ns
50 *	TdM1f(IORQf)	$\overline{M1}$ ↓ to $\overline{IORQ}$ ↓ Delay	365	—	270	—	ns
51	TdCf(IORQf)	Clock ↓ to $\overline{IORQ}$ ↓ Delay	—	70	—	60	ns
52	TdCr(IORQr)	Clock ↑ to $\overline{IORQ}$ ↑ Delay	—	70	—	60	ns
53	TdCf(D)	Clock ↓ to Data Valid Delay	—	130	—	115	ns

060489

Note 1 AC Test Condition
$V_{IH} = 2.4V$, $V_{IL} = 0.4V$, $V_{IHC} = V_{CC} - 0.6V$, $V_{ILC} = 0.6V$
$V_{OH} = 2.2V$, $V_{OL} = 0.8V$, $C_L = 100PF$

Note 2 Items with an asterisk (*) are non-compatible with NMOS Z80.

4.4 CAPACITANCE

$T_A = 25°C$

SYMBOL	ITEM	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
CCLOCK	Clock input Capacitance	f = 1MHz All pins except measured are connected to GND.	—	—	8	pF
CIN	Input Capacitance		—	—	6	pF
COUT	Output Capacitance		—	—	10	pF

060489

4.5 TIMING DIAGRAM

Figure 4.1 to 4.8 show the basic timings of respective operations. Numbers shown in the Figures correspond with those in the AC Electrical Characteristics Table in 4.3.

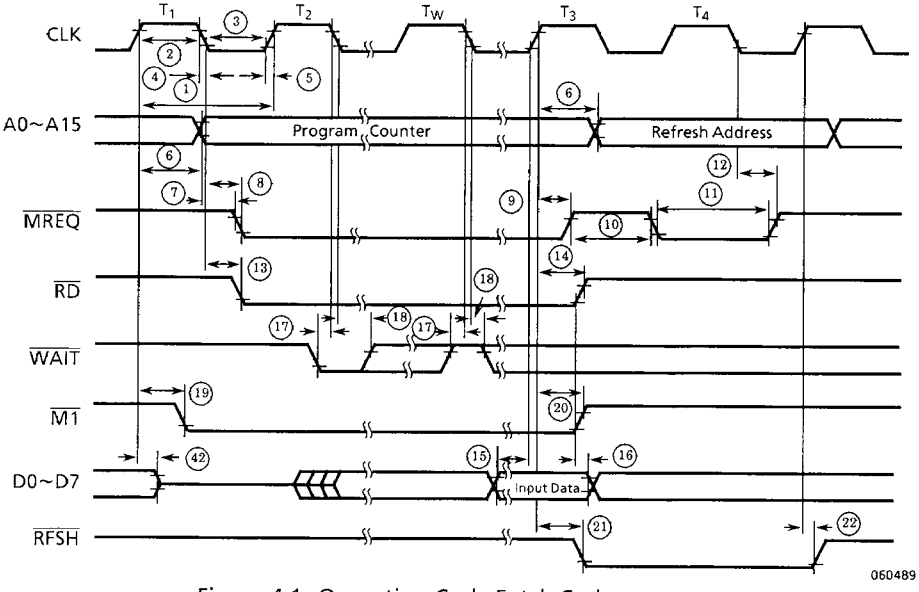


Figure 4.1 Operation Code Fetch Cycle

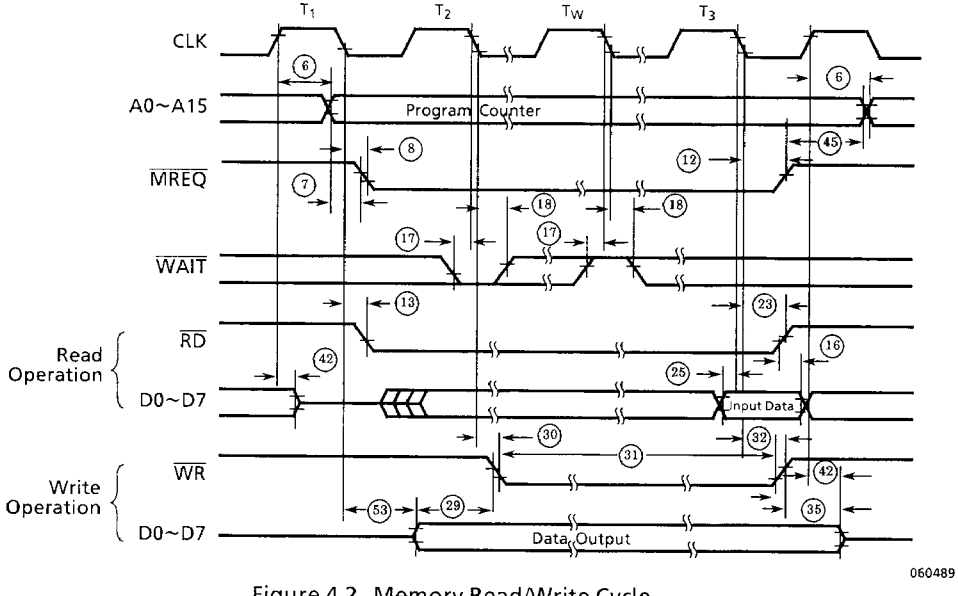
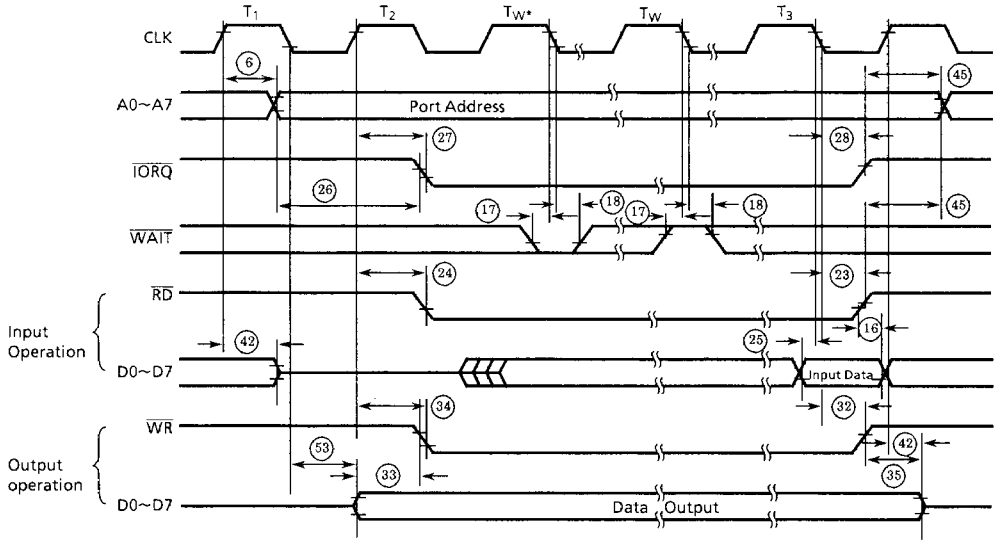
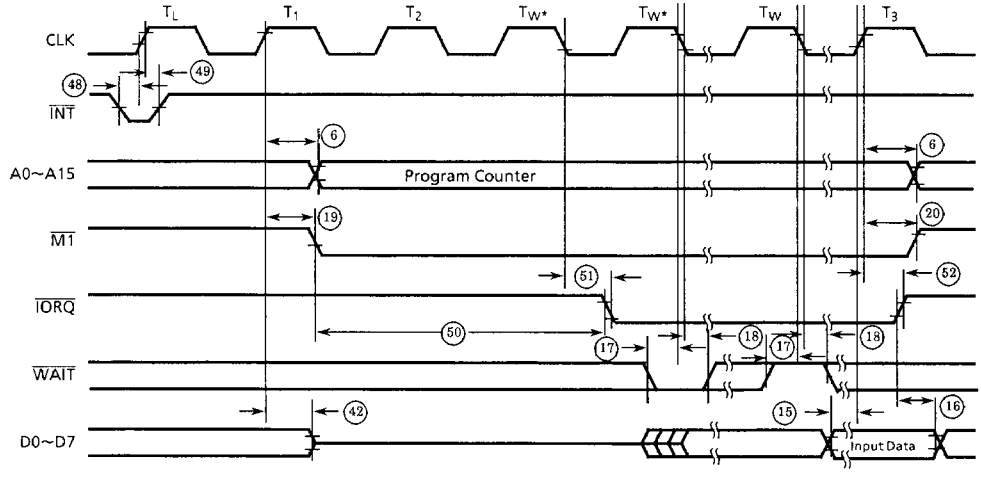


Figure 4.2 Memory Read/Write Cycle



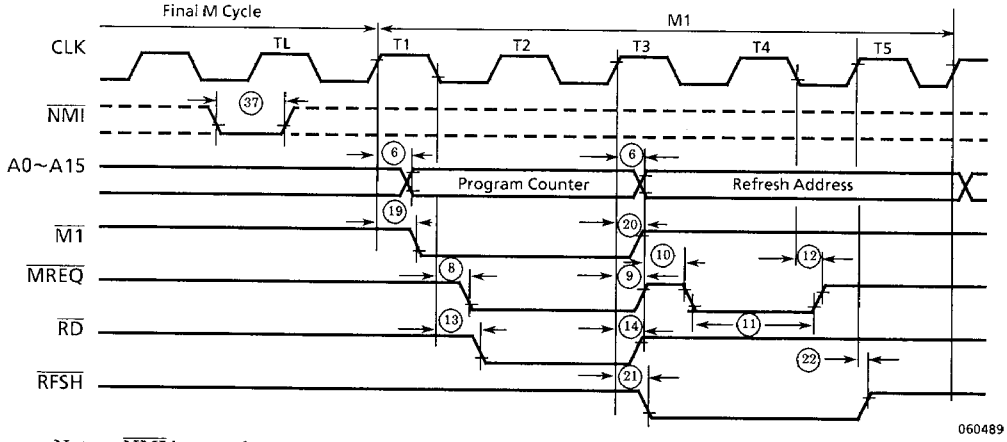
Note : 1 wait state (TW*) is inserted automatically by MPU. 060489

Figure 4.3 Input/Output Cycle



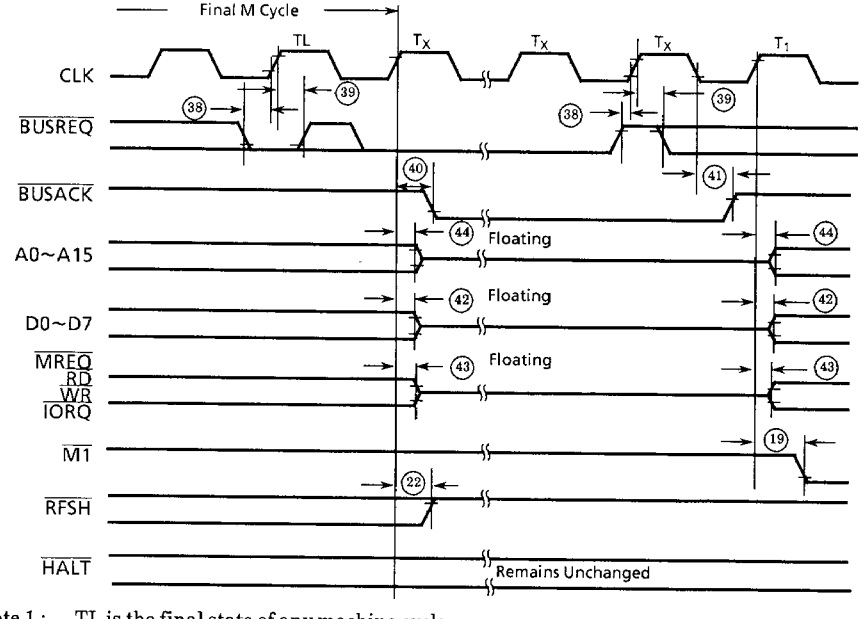
Note 1 TL is the final state of the preceding instruction.
Note 2 2 wait state (TW*) is inserted automatically by MPU. 060489

Figure 4.4 Interrupt Request/Acknowledge Cycle



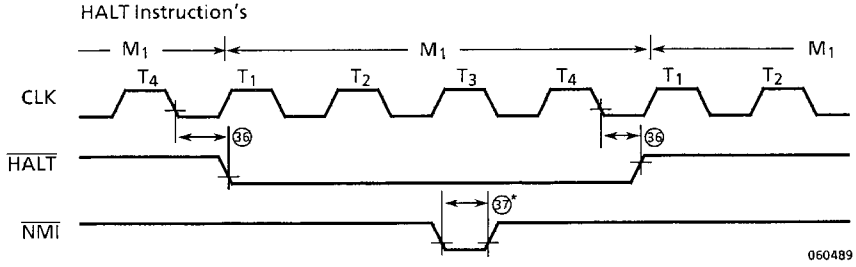
Note : $\overline{NMI}$ is asynchoronous input but in order to assure the positive response in the following cycle, $\overline{NMI}$ trailing edge signal must be generated keeping abreast of the leading edge of the preceding TL state.

Figure 4.5 Non-maskable Interrupt Request Cycle



Note 1 : TL is the final state of any machine cycle
Note 2 : TX is optional clock used by requested peripheral LSI.

Figure 4.6 Bus Request/Acknowledge Cycle



Note : $\overline{INT}$ signal is also used for releasing from the halt state

Figure 4.7 Halt Acknowledge Cycle

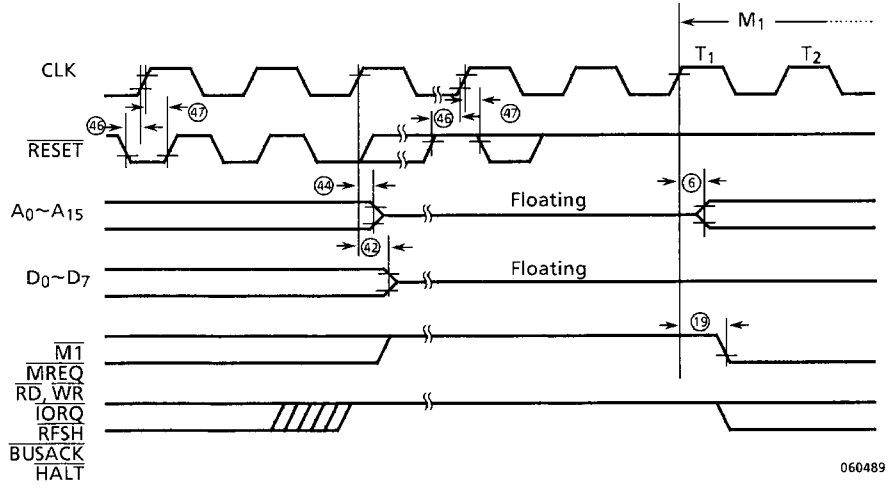


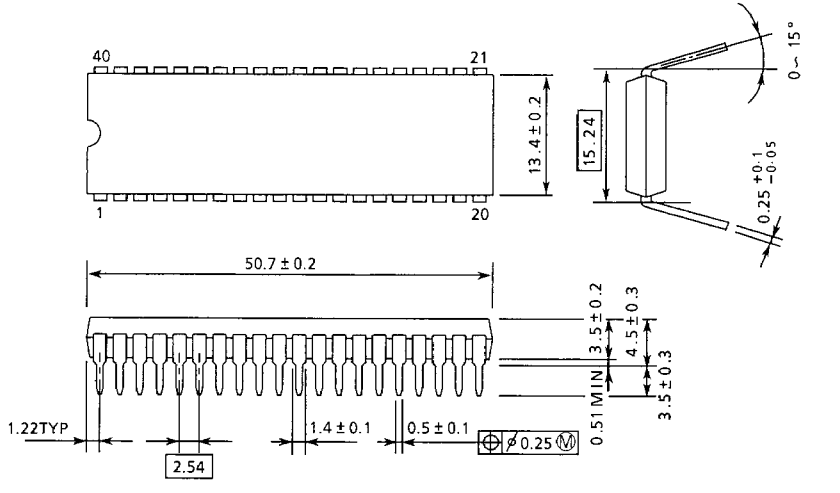
Figure 4.8 Reset Cycle

5. PACKAGE DIMENSION

5.1 DIP PACKAGE

DIP40-P-600

Unit : mm



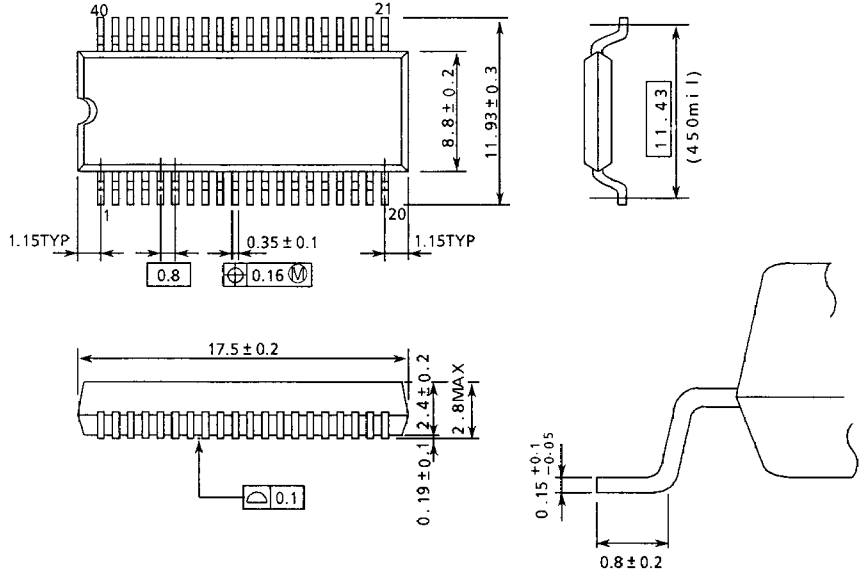
270289

Note 1 : This dimension is measured at the center of bending points of leads.
Note 2 : Each lead pitch is 2.54mm, and all the leads are located within ±0.25mm from their theoretical positions with respect to No.1 and No.40 leads.

5.2 SOP PACKAGE

SSOP40-P-450

Unit : mm



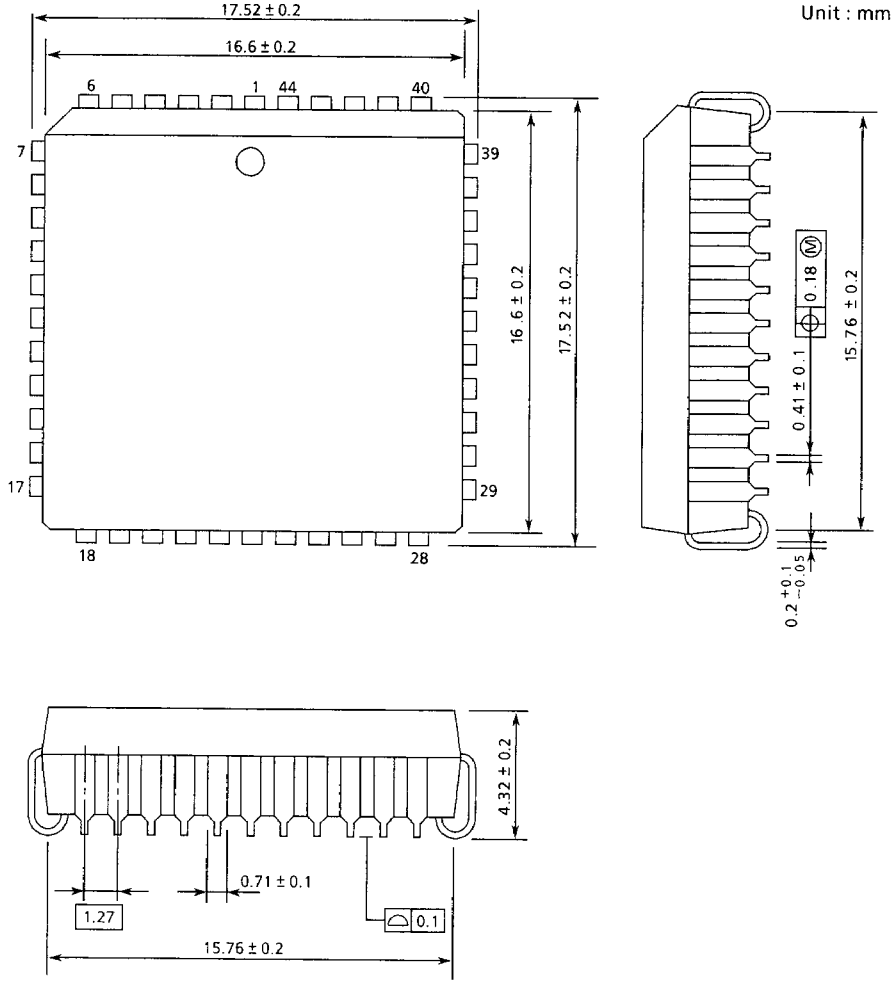
270289

Note : Package Width and length do not include Mold Protrusions.
Allowable Mold Protrusion is 0.15mm.

5.3 PLCC PACKAGE

QFJ44-P-S650

Unit : mm



270289

6. CAUTIONS

Please observe the following cautions when using the TMPZ8400A.

(1) The $\overline{RESET}$ signal input used for resetting must be held at "0" for at least 3 clocks.

(2) When the MPU is not the bus master ($\overline{BUSREQ}=0$), the memory is not refreshed because the $\overline{RFSH}$ signal is "1" and address signals are at high impedance. With systems using dynamic RAM, an external circuit is required for memory refresh if this condition persists for any length of time.

Also, interrupts cannot be received when the MPU is not the bus master.

(3) When exiting a power down operation with the MPU in hold status, supply the prescribed stabilized clock.

(4) Maskable interrupt mode 2 is only for use with Z80 family peripheral LSIs.

(5) Only the program counter, interrupt enable flip-flop, internal NMI flip-flop, I register and R register of the MPU are initialized. All other registers must be initialized by program when necessary. Also, set the interrupt mode to mode 0.

(6) The interrupt enable flip-flop is set to "1" by the instruction following the EI instruction to enable receipt of maskable interrupts.

(7) Only the program counter register is saved during interrupt processing. Save and restore interrupt processing routines as necessary.

(8) When using maskable interrupt mode 2, a data table for the vector addresses must be created in the memory.

(9) When periphral LSIs and memory are connected with the MPU on a PCB, use wiring as large as possible and the shortest routing for connecting Vss (GND) and Vcc.

Caution is necessary because of the large spike currents which can occur when signals change ($0\rightarrow1$, $1\rightarrow0$) with high-speed versions.

(10) As countermeasures for the above, connect a capacitor with good pulse response between Vcc and Vss (GND) of the MPU and other devices to absorb the pulse current.