DATA SHEET

MOS INTEGRATED CIRCUIT

μPD17107, 17107(A)

4 BIT SINGLE-CHIP MICROCONTROLLER

The μPD17107, tiny microcontroller, consists of 1K-byte (512 × 16 bits) ROM, 16 × 4 bit RAM, and 11 input/output ports.

The 17K architecture of the CPU uses general registers so that data memory can be manipulated directly for effective programming. Every instruction is 1 word long, consisting of 16 bits.

FEATURES

- Program memory (ROM) : 1K bytes (512 × 16 bits)
- Data memory (RAM) : 16 × 4 bits
- Input/output ports : 11 ports (including three N-ch open-drain outputs)
- Instruction execution time : 128 μs (for f_{CC} = 62.5 kHz) to 8 μs (for f_{CC} = 1 MHz)
- Stack level : 1
- A standby function (with the HALT and STOP modes)
- Data memory can retain data on low voltage (2.0 V at minimum).
- An RC oscillator for the system clock: Capacitors are built in (only a resistor is required externally).
- Operating supply voltage : 2.5 to 6.0 V (at f_{CC} = 250 kHz)
  4.5 to 6.0 V (at f_{CC} = 1 MHz)

APPLICATIONS

μPD17107 : Controlling electric appliances or toys
μPD17107(A) : Electronic units for automobiles, and such like

ORDERING INFORMATION

Part number	Package	Quality grade	
μPD17107CX-×××	16-pin plastic DIP (300 mil)	Standard	
μPD17107GS-×××	16-pin plastic SOP (300 mil)	Standard	
μPD17107CX(A)-×××	16-pin plastic DIP (300 mil)	Special	★
μPD17107GS(A)-×××	16-pin plastic SOP (300 mil)	Special	★

Remark ××× : ROM code number

The only difference between the μPD17107 and μPD17107(A) is the quality grade. Only the μPD17107 is described in this manual. The description is also applicable to the μPD17107(A), however.

Please refer to "Quality grade on NEC Semiconductor Devices" (Document number IEI-1209) published by NEC Corporation to know the specification of quality grade on the devices and its recommended applications.

The information in this document is subject to change without notice.

Major changes in this revision are indicated by stars (★) in the margins.

Document No. IC-2643C
(O.D. No. IC-8060C)
Date Published May 1995 P
Printed in Japan

© NEC Corporation 1991

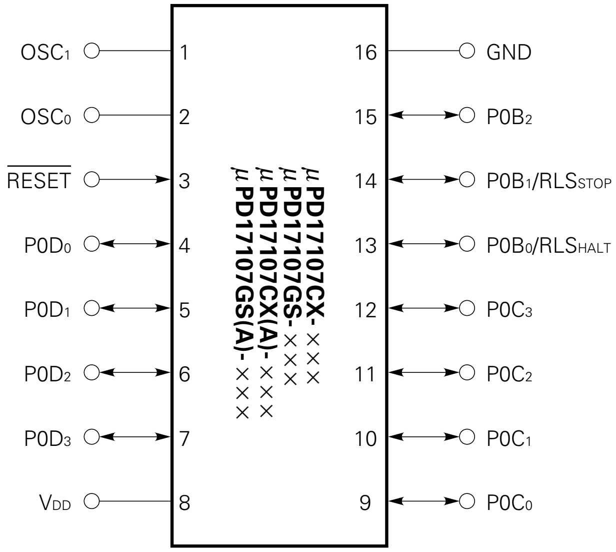
★ FUNCTIONS

Item	Function
ROM	1K byte (512 × 16 bits)
RAM	16 × 4 bits
Stack	1 level
Number of I/O ports	11 (N-ch open-drain output ports: 3)
System clock (f_{CC})	RC oscillation
Instruction execution time	128 μs (when f_{CC} = 62.5 kHz) to 8 μs (when f_{CC} = 1 MHz)
Standby function	HALT/STOP
Operating supply voltage	2.5 to 6.0 V (at f_{CC} = 62.5 kHz to 250 kHz) 4.5 to 6.0 V (at f_{CC} = 62.5 kHz to 1 MHz)
Package	16-pin plastic DIP (300 mil) 16-pin plastic SOP (300 mil)
One-time PROM	μPD17P107

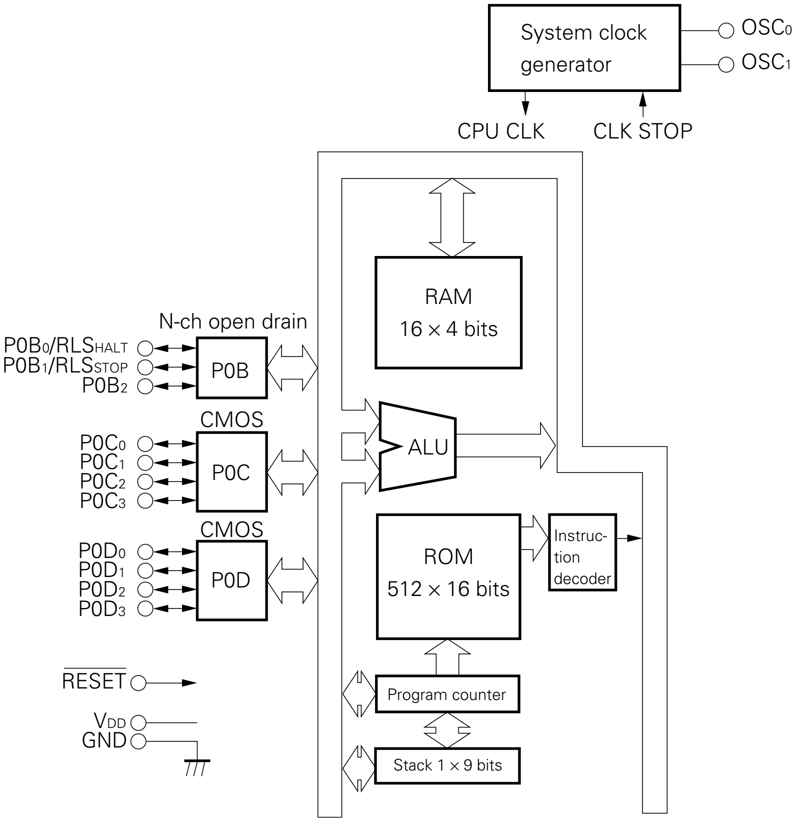
Caution Although a PROM product is highly compatible with a masked ROM product in respect of functions, they differ in internal ROM circuits and part of electrical characteristics. Before changing the PROM product to the masked ROM product in an application system, evaluate the system carefully using the masked ROM product.

PIN CONFIGURATION (TOP VIEW)

16-pin plastic DIP
16-pin plastic SOP



BLOCK DIAGRAM



PINS

Pin functions

- Port pins

Pin	I/O	Function	Reset
$P0B_0/RLS_{HALT}$	I/O	For releasing the HALT mode	• Open-drain: High impedance (input mode) • With pull-up resistor selected: High Level (input mode)
$P0B_1/RLS_{STOP}$		For releasing the STOP mode	
$P0B_2$		• N-ch open-drain 3-bit I/O port (port 0B) • A built-in pull-up resistor can be connected with a mask option bit by bit. • This open-drain port has a withstand voltage of 9 V.	
$P0C_0$-$P0C_3$	I/O	CMOS (push-pull) 4-bit I/O port (port 0C)	High impedance (input mode)
$P0D_0$-$P0D_3$	I/O	CMOS (push-pull) 4-bit I/O port (port 0D)	High impedance (input mode)

- Non-port pins

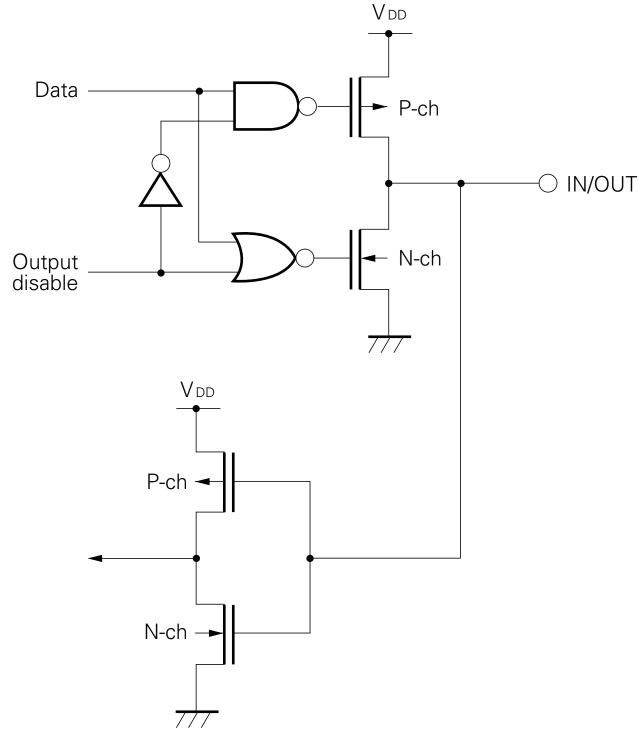
Pin	I/O	Function
$\overline{\text{RESET}}$	Input	• Reset input pin • A built-in pull-up resistor can be connected with a mask option.
V_{DD}	–	Positive power supply pin
GND	–	GND pin
OSC_0, OSC_1	–	Pins through which a resistor is connected to the system clock resonator

I/O: Input/output

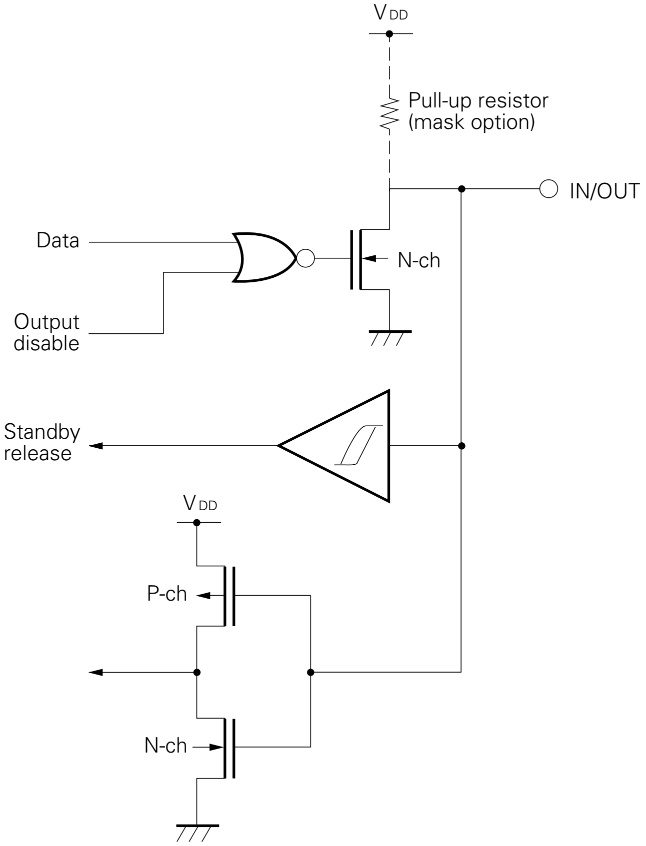
Equivalent input/output circuits

Below are simplified diagrams of the equivalent input/output circuits.

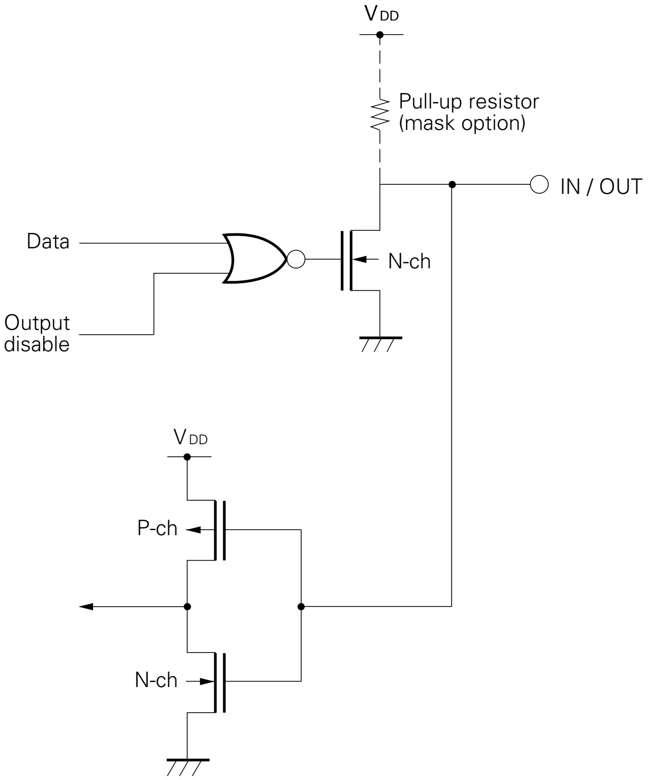
(1) P0C and P0D



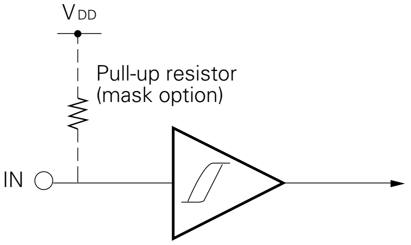
(2) P0B0 and P0B1



(3) P0B₂



(4) $\overline{\text{RESET}}$



HANDLING UNUSED PINS

★

When connecting unused pins, the following conditions and handling are recommended:

Pin			Recommended conditions and handling	
			Internal	External
Port	Input mode	P0C, P0D	–	Connect each pin to V_{DD} or to ground through a resistor[Note].
		P0B	Pull-up resistors that can be specified by the mask option are not incorporated.	
			Pull-up resistors that can be specified with the mask option are incorporated.	Leave open.
	Output mode	P0C, P0D (CMOS ports)	–	
		P0B (N-ch open-drain port)	Outputs low level without pull-up resistors that can be specified with the mask option	
			Outputs high level with pull-up resistors that can be specified with the mask option	

Note When a pin is pulled up (connected to V_{DD} through a resistor) or pulled down (connected to ground through a resistor) outside the chip, take the driving capacity and maximum current consumption of a port into consideration. When using high-resistance pull-up or pull-down resistors, apply appropriate countermeasures to ensure that noise is not attracted by the resistors. Although the optimum pull-up or pull-down resistor varies with the application circuit, in general a resistor of 10 to 100 kilohms is suitable.

Caution To fix the output level of a pin, it is recommended that the level be specified repeatedly within a loop in a program.

NOTES ON USE OF THE $\overline{\text{RESET}}$ PIN

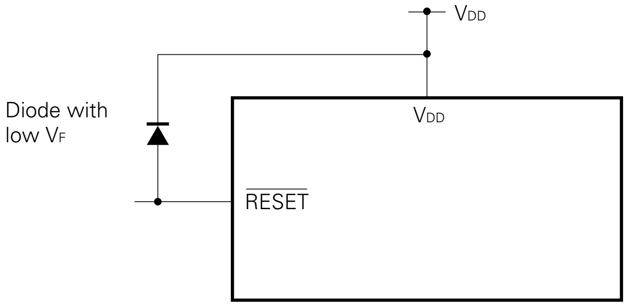
The $\overline{\text{RESET}}$ pin has the test mode selecting function for testing the internal operation of the μPD17107 (IC test), besides the functions shown in " **PINS**".

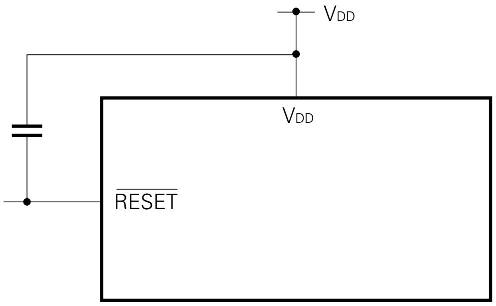
Applying a voltage exceeding V_{DD} to the $\overline{\text{RESET}}$ pin causes the μPD17107 to enter the test mode. When noise exceeding V_{DD} comes in during normal operation, the device is switched to the test mode.

For example, if the wiring from the $\overline{\text{RESET}}$ pin is too long, noise may be induced on the wiring, causing this mode switching.

When installing the wiring, lay the wiring in such a way that noise is suppressed as much as possible. If noise yet arises, use an external part to suppress it as shown below.

- **Connect a diode with low V_F between the pin and V_{DD}.**
- **Connect a capacitor between the pin and V_{DD}.**





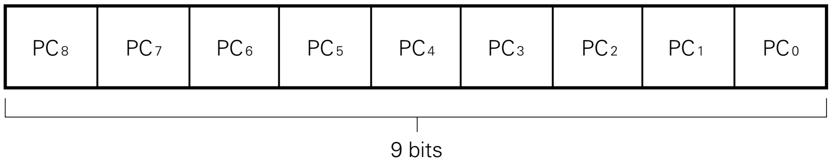
CONTENTS

1. PROGRAM COUNTER (PC)

1.1 CONFIGURATION OF THE PROGRAM COUNTER (PC)

As shown in Fig. 1-1, the program counter is a 9-bit binary counter.

Fig. 1-1 Program Counter



1.2 FUNCTIONS OF THE PROGRAM COUNTER (PC)

The program counter specifies the address of a program memory (ROM) or a program.

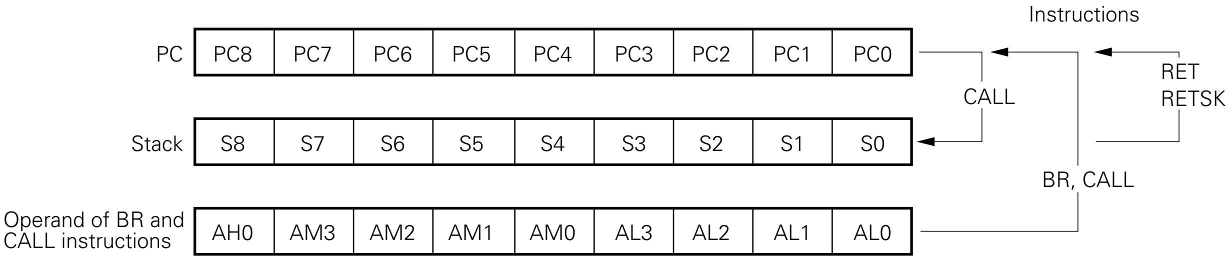
Usually, every time an instruction is executed, the program counter is incremented by one. When a branch instruction (BR), a subroutine call instruction (CALL), or a return instruction (RET) is executed, the address specified in the operand is loaded in the PC. Then the instruction in the address is executed. When a skip instruction is executed, the address of the instruction next to the skip instruction is specified irrespective of the contents of the skip instruction. If the skip conditions are satisfied, the instruction next to the skip instruction is regarded as a No Operation (NOP) instruction. So, the NOP instruction is executed and the address of the next instruction is specified.

2. STACK

Stack of the μPD17107 is a register in which the return address of a program is saved when a subroutine call instruction is executed. One level of address stack is provided.

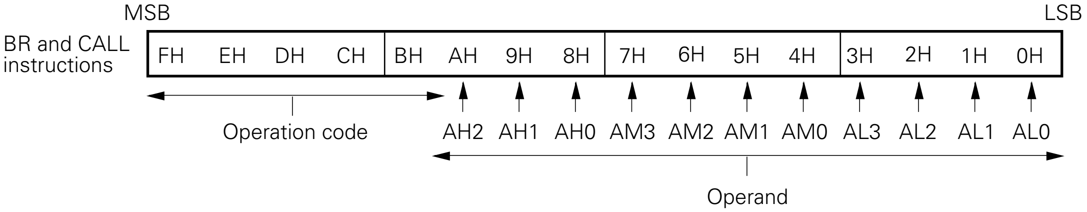
Fig. 2-1 shows the relationship between the PC, the stack, and the operand of BR and CALL instructions.

Fig. 2-1 Relationship between the PC, the Stack, and the Operand of BR and CALL Instructions



In Fig. 2-1, AHn, AMn, and ALn (n = 0 to 3) indicate bit positions in a 16-bit instruction as follows:

Fig. 2-2 Configuration of a 16-Bit Instruction



When the assembler (AS17K) is not used and a BR or CALL instruction is used, AH2 and AH1 must be set to 0.

Reset input clears all bits of the program counter to 0.

3. PROGRAM MEMORY (ROM)

Fig. 3-1 shows the program memory (ROM) configuration.

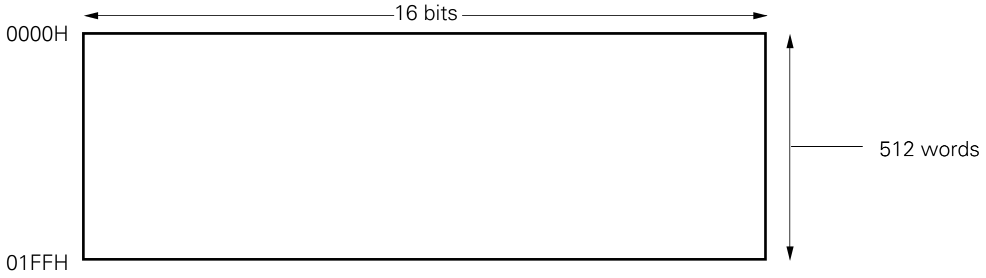
As shown in the figure, the program memory has 512 words by 16 bits.

The program memory has been addressed in units of 16 bits. The addresses 0000H to 01FFH are specified by the program counter (PC).

Every instruction is a 1 word long, consisting of 16 bits. One instruction can therefore be stored at one address in program memory.

Address 0000H is used as a reset start address.

Fig. 3-1 Program Memory Map



4. DATA MEMORY (RAM)

The data memory stores data of arithmetic/logic and control operations. Data can be always written to or read from it by means of instructions.

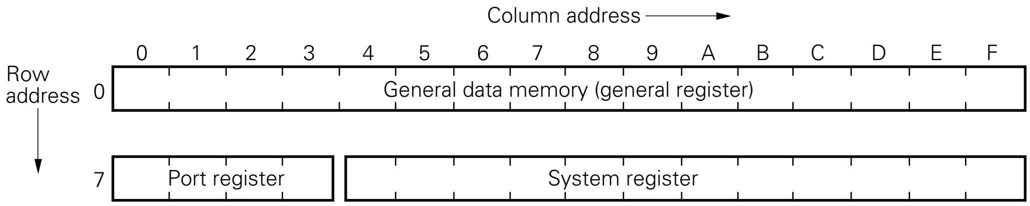
4.1 CONFIGURATION OF THE DATA MEMORY (RAM)

Fig. 4-1 shows the configuration of the data memory (RAM).

The data memory is configured in units of four bits, or "one nibble," and an address is assigned to each four bits of data. The high-order three bits are called the "row address," and the low-order four bits are called the "column address."

According to its functions, the data memory is divided into three blocks as shown below: General data memory, port register, and system register.

Fig. 4-1 Data Memory Map



4.1.1 Functions of the General Data Memory

The general data memory is a part of the data memory from which the system register (SYSREG) and port register are excluded. By executing a data memory manipulation instruction, a four-bit arithmetic operation and comparison, evaluation, and transfer between data on data memory and any immediate data can be executed with a single operation.

4.1.2 Functions of the General Register

The general register indicates any identical row address (16 nibbles) in the data memory specified in the register pointer (RP) in the system register. Since the μPD17107 register pointer is always set to 0, the general data memory is also used as a general register. The general register can operate or transfer data to and from the data memory.

4.1.3 Functions of the Port Register

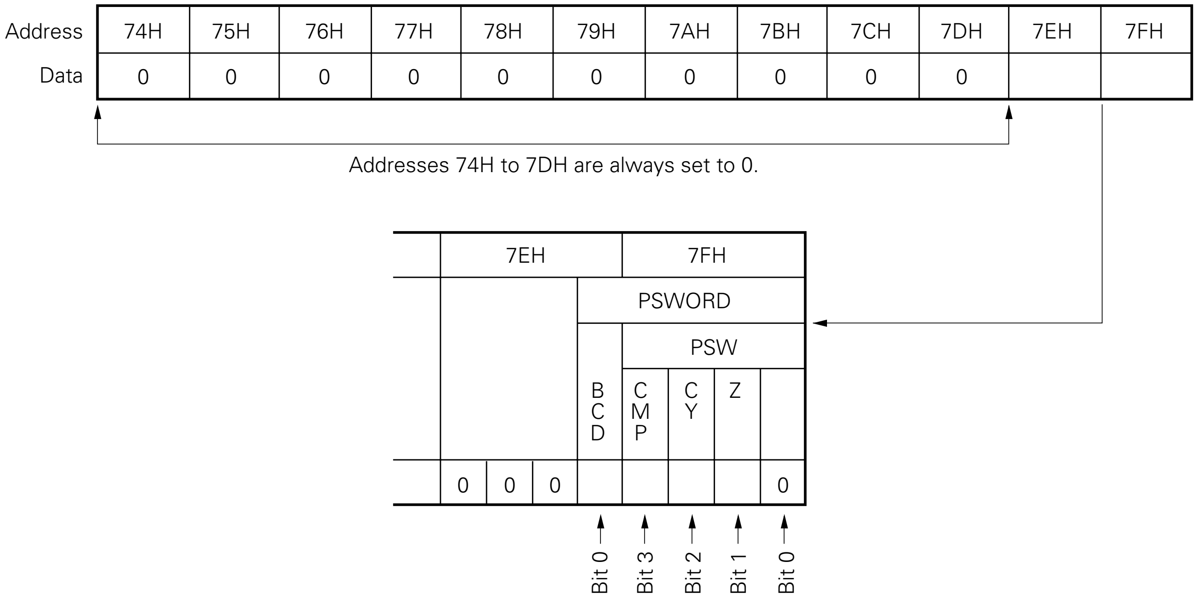
The port register is used to set output data or to read the input data of input/output ports.

Once data is written to the port register corresponding to a port, the port is set to output mode and outputs the data unless another data is rewritten (the output mode is maintained until the port register is reset). Whenever a read instruction is executed for a port register, the read data indicates the states of the pins, not the value of the port register, regardless of whether the pins are in the input or output mode.

4.1.4 Functions of the System Register

The system register controls the CPU. The program status word (PSWORD) is the only system register existing in the μPD17107.

★ **Fig. 4-2 System Register Map**

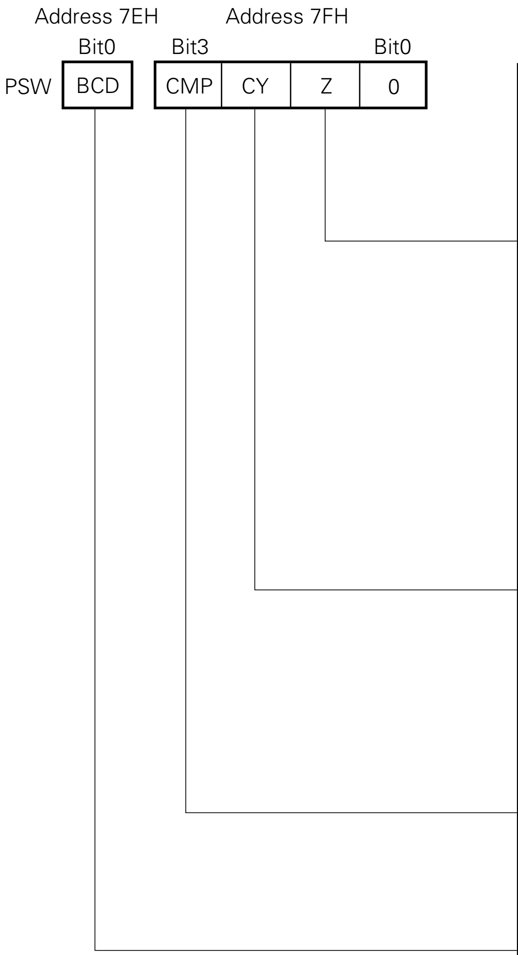


Bit 0 at address 7EH and the four bits at address 7FH (PSW) are assigned to the program status word.

The BCD flag is mapped in bit 0 at address 7EH, the CMP flag is mapped in bit 3 at address 7FH, the CY flag is mapped in bit 2, and the Z flag is mapped in bit 1 at address 7FH.

The high-order three bits at address 7EH and bit 0 at address 7FH are always set to 0.

Fig. 4-3 Configuration of the Program Status Word



Zero flag (Z)	Set to 1 when: • An arithmetic operation generates a result of zero if CMP = 0. • An arithmetic operation generates a result of zero and Z = 1 if CMP = 1. When the Z flag is already 0, it remains unchanged. Reset to 0 when: • An arithmetic operation generates a result other than zero.
Carry flag (CY)	Set to 1 when: • An addition produces a carry or a subtraction produces a borrow. • The LSB of the operand in the RORC instruction is 1. Reset to 0 when: • Neither a carry nor borrow is produced. • The LSB of the operand in the RORC instruction is 0.
Compare flag (CMP)	If this flag is set, the result of an arithmetic operation is not stored in memory or general registers. The flag is automatically reset by executing the SKT or SKF instruction.
BCD flag (BCD)	If this flag is set, arithmetic operations are performed in decimal, and if this flag is reset, arithmetic operations are performed in binary.

Comparison instructions (SKE, SKNE, SKGE, or SKLT) do not change the state of the CY flag, but an arithmetic operation may affect the CY flag according to the result even if the CMP flag is set.

Each bit of the program status word is initialized to 0 when a reset signal is applied.

The Z flag in the program status word changes according to the set value of the CMP flag as listed in Table 4-1.

★ **Table 4-1 Change in Z Flag**

Conditions	CMP = 0	CMP = 1
When arithmetic operation results in 0	Z ← 1	Z flag does not change
When arithmetic operation results in a non-zero value	Z ← 0	Z ← 0

While CMP is 1, if an arithmetic operation results in 0H when the value of the Z flag is 1, the Z flag does not change. If an arithmetic operation results in other than 0H, the Z flag is reset to 0 and remains intact even when a second arithmetic operation results in 0H.

After the CMP and Z flags are set to 1, subtraction and comparison are performed several times. Then, if the Z flag still indicates 1, all of the comparison operations showed a match, resulting in 0. If the Z flag is 0 after the comparison operations, a mismatch occurred in at least one comparison operation.

Example of 12-bit data comparison

```
; Is the 12-bit data stored in M001, M002, and M003 equal to 456H?
CMP456:
          SET2    CMP, Z
          SUB     M001, #4      ; Stores the data in M001, M002, and M003.
          SUB     M002, #5      ; Does not damaged the data.
          SUB     M003, #6      ;
          ;CLR1   CMP
          SKT1    Z             ; Resets CMP automatically when the bit test instruction is executed.
          BR      DIFFER        ; ≠ 456H
          BR      AGREE         ; = 456H
```

5. ALU BLOCK

★

The ALU is used for performing arithmetic operations, logical operations, bit evaluations, comparison evaluations, and rotations on 4-bit data.

5.1 ALU BLOCK CONFIGURATION

Fig. 5-1 shows the configuration of the ALU block.

As shown in Fig. 5-1, the ALU block consists of the main 4-bit data processor, temporary registers A and B, the status flip-flop for controlling the status of the ALU, and the decimal conversion circuit for use during arithmetic operations in BCD.

As shown in Fig. 5-1, the status flip-flop consists of the following flags: Zero flag flip-flop, carry flag flip-flop, compare flag flip-flop, and the BCD flag flip-flop.

Each flag in the status flip-flop corresponds directly to a flag in the program status word (PSWORD: addresses 7EH, 7FH) located in the system register. The flags in the program status word are the following: Zero flag (Z), carry flag (CY), compare flag (CMP), and the BCD flag (BCD).

5.2 FUNCTIONS OF THE ALU BLOCK

Arithmetic operations, logical operations, bit evaluations, comparison evaluations, and rotations are performed using the instructions in the ALU block. Table 5-1 lists each arithmetic/logical instruction, evaluation instruction, and rotation instruction.

By using the instructions listed in Table 5-1, 4-bit arithmetic/logical operations, evaluations and rotations can be performed in a single instruction. Arithmetic operations in BCD can also be performed on one place in a single instruction.

5.2.1 Functions of the ALU

The arithmetic operations consist of addition and subtraction. Arithmetic operations can be performed on the contents of the general register and data memory or on immediate data and the contents of data memory. Operations in binary are performed on four bits of data and operations in BCD are performed on one place.

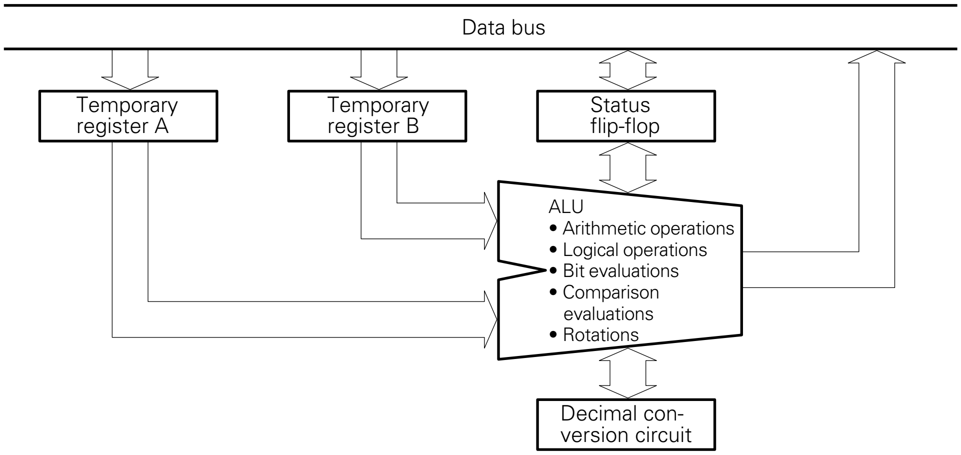
Logical operations include ANDing, ORing, and XORing. Their operands can be general register contents and data memory contents, or data memory contents and immediate data.

Bit evaluation is used to determine whether bits in 4-bit data in data memory are 0 or 1.

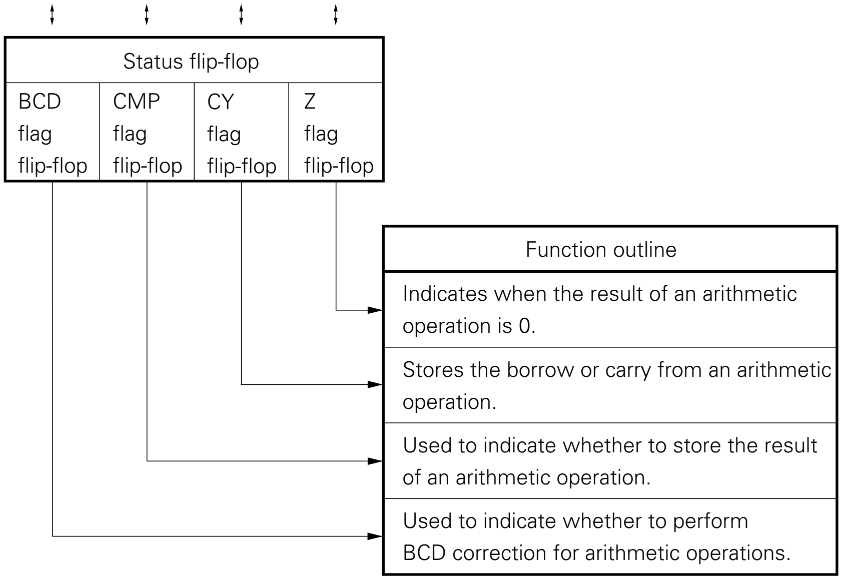
Comparison evaluation is used to compare contents of data memory with immediate data. It is used to determine whether one value is equal to or greater than the other, less than the other, or if both values are equal or not equal.

Rotation is used to shift 4-bit data in the general register one bit in the direction of its least significant bit (rotation to the right).

Fig. 5-1 Configuration of the ALU



Address	7EH	7FH			
Name	Program status word (PSWORD)				
Bit	b_0	b_3	b_2	b_1	b_0
Flag	BCD	CMP	CY	Z	0



Function outline
Indicates when the result of an arithmetic operation is 0.
Stores the borrow or carry from an arithmetic operation.
Used to indicate whether to store the result of an arithmetic operation.
Used to indicate whether to perform BCD correction for arithmetic operations.

[MEMO]

Table 5-1 List of ALU Instructions (1/2)

ALU function		Instruction	Operation	Explanation
Arithmetic operations	Addition	ADD r, m	(r) ← (r) + (m)	Adds contents of general register and data memory. Result is stored in general register.
		ADD m, #n4	(m) ← (m) + n4	Adds immediate data to contents of data memory. Result is stored in data memory.
		ADDC r, m	(r) ← (r) + (m) + CY	Adds contents of general register, data memory and carry flag. Result is stored in general register.
		ADDC m, #n4	(m) ← (m) + n4 + CY	Adds immediate data, contents of data memory and carry flag. Result is stored in data memory.
	Subtraction	SUB r, m	(r) ← (r) - (m)	Subtracts contents of data memory from contents of general register. Result is stored in general register.
		SUB m, #n4	(m) ← (m) - n4	Subtracts immediate data from data memory. Result is stored in data memory.
		SUBC r, m	(r) ← (r) - (m) - CY	Subtracts contents of data memory and carry flag from contents of general register. Result is stored in general register.
		SUBC m, #n4	(m) ← (m) - n4 - CY	Subtracts immediate data and carry flag from data memory. Result is stored in data memory.
Logical operations	Logical OR	OR r, m	(r) ← (r) ∨ (m)	OR operation is performed on contents of general register and data memory. Result is stored in general register.
		OR m, #n4	(m) ← (m) ∨ n4	OR operation is performed on immediate data and contents of data memory. Result is stored in data memory.
	Logical AND	AND r, m	(r) ← (r) ∧ (m)	AND operation is performed on contents of general register and data memory. Result is stored in general register.
		AND m, #n4	(m) ← (m) ∧ n4	AND operation is performed on immediate data and contents of data memory. Result is stored in data memory.
	Logical XOR	XOR r, m	(r) ← (r) ∀ (m)	XOR operation is performed on contents of general register and data memory. Result is stored in general register.
		XOR m, #n4	(m) ← (m) ∀ n4	XOR operation is performed on immediate data and contents of data memory. Result is stored in data memory.
Bit evaluation	True	SKT m, #n	CMP←0, if (m) ∧ n = n, then skip	Skips next instruction if all bits in data memory specified by n are TRUE (1). Result is not stored.
	False	SKF m, #n	CMP←0, if (m) ∧ n = 0, then skip	Skips next instruction if all bits in data memory specified by n are FALSE (0). Result is not stored.
Comparison evaluation	Equal	SKE m, #n4	(m) - n4, skip if zero	Skips next instruction if immediate data equals contents of data memory. Result is not stored.
	Not equal	SKNE m, #n4	(m) - n4, skip if not zero	Skips next instruction if immediate data is not equal to contents of data memory. Result is not stored.
	≧	SKGE m, #n4	(m) - n4, skip if not borrow	Skips next instruction if contents of data memory is greater than or equal to immediate data. Result is not stored.
	<	SKLT m, #n4	(m) - n4, skip if borrow	Skips next instruction if contents of data memory is less than immediate data. Result is not stored.
Rotation	Rotate to the right	RORC r	$\rightarrow (CY) \rightarrow (r)_{b3} \rightarrow (r)_{b2} \rightarrow (r)_{b1} \rightarrow (r)_{b0}$	Rotate contents of the general register along with the CY flag to the right. Result is stored in general register.

Table 5-1 List of ALU Instructions (2/2)

ALU function	Operation depending on the program status word (PSWORD)				
	Value in BCD flag	Value in CMP flag	Operation	CY flag	Z flag
Arithmetic operation	0	0	Store result of binary operation	Set (1) when carry or borrow is generated, otherwise reset (0).	Set (1) when result of operation is 0000B, otherwise reset (0).
	0	1	Do not store result of binary operation		Status maintained when result of operation is 0000B, otherwise reset (0).
	1	0	Store result of decimal operation		Set (1) when result of operation is 0000B, otherwise reset (0).
	1	1	Do not store result of decimal operation		Status maintained when result of operation is 0000B, otherwise reset (0).
Logical operations	Don't care (maintained)	Don't care (maintained)	No change	Don't care (maintained)	Don't care (maintained)
Bit evaluation	Don't care (maintained)	Reset	No change	Don't care (maintained)	Don't care (maintained)
Comparison evaluation	Don't care (maintained)	Don't care (maintained)	No change	Don't care (maintained)	Don't care (maintained)
Rotation	Don't care (maintained)	Don't care (maintained)	No change	Value in b_0 of the general register	Don't care (maintained)

5.2.2 Functions of Temporary Registers A and B

Temporary registers A and B are needed for processing of 4-bit data. These registers are used for temporary storage of the first and second data operands of an instruction.

5.2.3 Functions of the Status Flip-flop

The status flip-flop is used for controlling operation of the ALU and for storing data which has been processed. Each flag in the status flip-flop corresponds directly to a flag in the program status word (PSWORD) located in the system register. This means that when a flag in the system register is manipulated it is the same as manipulating a flag in the status flip-flop. Each flag in the program status word is described below.

(1) Z flag

This flag is set (1) when the result of an arithmetic operation is 0000B, otherwise it is reset (0). However, as described below, depending on the status of the CMP flag, the conditions which cause this flag to be set (1) can be changed.

(i) When CMP = 0

Z flag is set (1) when the result of an arithmetic operation is 0000B, otherwise it is reset (0).

(ii) When CMP = 1

The previous state of the Z flag is maintained when the result of an arithmetic operation is 0000B, otherwise it is reset (0). Only affected by arithmetic operations.

(2) CY flag

This flag is set (1) when a carry or borrow is generated in the result of an arithmetic operation, otherwise it is reset (0).

When an arithmetic operation is being performed using a carry or borrow, the operation is performed using the CY flag as the least significant bit. When a rotation (RORC instruction) is performed, the contents of the CY flag becomes the most significant bit (bit b_3) of the general register and the least significant bit of the general register is stored in the CY flag.

Only affected by arithmetic operations and rotations.

(3) CMP flag

When the CMP flag is set (1), the result of an arithmetic operation is not stored in either the general register or data memory.

When the bit evaluation instruction is performed, the CMP flag is reset (0).

The CMP flag does not affect comparison evaluations, logical operations, or rotations.

(4) BCD flag

When the BCD flag is set (1), all arithmetic operations are performed in BCD. When the flag is reset (0), all operations are performed in 4-bit binary.

The BCD flag does not affect logical operations, bit evaluations, comparison evaluations, or rotations.

These flags can also be set through direct manipulation of the values in the program status word (PSWORD). When the flags in the program status word are manipulated, the corresponding flag in the status flip-flop is also manipulated.

5.2.4 Performing Operations in 4-Bit Binary

When the BCD flag is set to 0, arithmetic operations are performed in 4-bit binary.

5.2.5 Performing Operations in BCD

When the BCD flag is set to 1, arithmetic operations are performed in BCD. Table 5-2 shows the differences in the results of operations performed in 4-bit binary and in BCD. When the result of an addition in BCD is equal to or greater than 20, or the result of a subtraction in BCD is outside of the range -10 to +9, a value of 1010B (0AH) or higher is stored as the result (shaded area in Table 5-2).

Table 5-2 Results of Arithmetic Operations Performed in 4-Bit Binary and BCD

Operation result	Addition in 4-bit binary		Addition in BCD	
	CY	Operation result	CY	Operation result
0	0	0000	0	0000
1	0	0001	0	0001
2	0	0010	0	0010
3	0	0011	0	0011
4	0	0100	0	0100
5	0	0101	0	0101
6	0	0110	0	0110
7	0	0111	0	0111
8	0	1000	0	1000
9	0	1001	0	1001
10	0	1010	1	0000
11	0	1011	1	0001
12	0	1100	1	0010
13	0	1101	1	0011
14	0	1110	1	0100
15	0	1111	1	0101
16	1	0000	1	0110
17	1	0001	1	0111
18	1	0010	1	1000
19	1	0011	1	1001
20	1	0100	1	1110
21	1	0101	1	1111
22	1	0110	1	1100
23	1	0111	1	1101
24	1	1000	1	1110
25	1	1001	1	1111
26	1	1010	1	1100
27	1	1011	1	1101
28	1	1100	1	1010
29	1	1101	1	1011
30	1	1110	1	1100
31	1	1111	1	1101

Operation result	Subtraction in 4-bit binary		Subtraction in BCD	
	CY	Operation result	CY	Operation result
0	0	0000	0	0000
1	0	0001	0	0001
2	0	0010	0	0010
3	0	0011	0	0011
4	0	0100	0	0100
5	0	0101	0	0101
6	0	0110	0	0110
7	0	0111	0	0111
8	0	1000	0	1000
9	0	1001	0	1001
10	0	1010	1	1100
11	0	1011	1	1101
12	0	1100	1	1110
13	0	1101	1	1111
14	0	1110	1	1100
15	0	1111	1	1101
-16	1	0000	1	1110
-15	1	0001	1	1111
-14	1	0010	1	1100
-13	1	0011	1	1101
-12	1	0100	1	1110
-11	1	0101	1	1111
-10	1	0110	1	0000
-9	1	0111	1	0001
-8	1	1000	1	0010
-7	1	1001	1	0011
-6	1	1010	1	0100
-5	1	1011	1	0101
-4	1	1100	1	0110
-3	1	1101	1	0111
-2	1	1110	1	1000
-1	1	1111	1	1001

5.2.6 Performing Operations in the ALU Block

When arithmetic operations, logical operations, bit evaluations, comparison evaluations or rotations in a program are executed, the first data operand is stored in temporary register A and the second data operand is stored in temporary register B.

The first data operand is four bits of data used to specify the contents of an address in the general register or data memory. The second data operand is four bits of data used to either specify the contents of an address in data memory or to be used as an immediate value. For example, in the instruction

ADD r, m
Second data operand
First data operand

the first data operand, r, is used to specify the contents of an address in the general register. The second data operand, m, is used to specify the contents of an address in data memory. In the instruction

ADD m, #n4

the first data operand, m, is used to specify an address in data memory. The second operand, #n4, is immediate data. In the rotation instruction

RORC r

only the first data operand, r (used to specify the contents of an address in the general register) is used.

Next, using the data stored in temporary registers A and B, the ALU executes the operation specified by the instruction (arithmetic operation, logical operation, bit evaluation, comparison evaluation, or rotation). When the instruction being executed is an arithmetic operation, logical operation, or rotation, the data processed by the ALU is stored in the location specified by the first data operand (general register address or data memory address) and the operation terminates. When the instruction being executed is a bit evaluation or comparison evaluation, the result processed by the ALU is used to determine whether or not to skip the next instruction (whether to treat next instruction as a no operation instruction: NOP) and the operation terminates.

Caution should be taken with regard to the following points:

(1) Arithmetic operations are affected by the CMP and BCD flags in the program status word.
(2) Logical operations are not affected by the CMP or BCD flag in the program status word. Logical operations do not affect the Z or CY flags.
(3) Bit evaluation causes the CMP flag in the program status word to be reset.

5.3 ARITHMETIC OPERATIONS (ADDITION AND SUBTRACTION IN 4-BIT BINARY AND BCD)

As shown in Table 5-3, arithmetic operations consist of addition, subtraction, addition with carry, and subtraction with borrow. These instructions are ADD, ADDC, SUB, and SUBC.

The ADD, ADDC, SUB, and SUBC instructions are further divided into addition and subtraction of the general register and data memory and addition and subtraction of data memory and immediate data. When the operands r and m are used, addition or subtraction is performed using the general register and data memory. When the operands m and #n4 are used, addition or subtraction is performed using data memory and immediate data.

Arithmetic operations are affected by the status flip-flop and the program status word (PSWORD) in the system register. The BCD flag in the program status word (PSWORD) is used to specify whether arithmetic operations are to be performed in 4-bit binary or in BCD. The CMP flag is used to specify whether or not the results of arithmetic operations are to be stored.

Sections 5.3.1 to **5.3.4** explain the relationship between each command and the program status word (PSWORD).

Table 5-3 Types of Arithmetic Operations

Arithmetic operation	Addition	Without carry ADD	General register and data memory	ADD r, m
			Data memory and immediate data	ADD m, #n4
		With carry ADDC	General register and data memory	ADDC r, m
			Data memory and immediate data	ADDC m, #n4
	Subtraction	Without borrow SUB	General register and data memory	SUB r, m
			Data memory and immediate data	SUB m, #n4
		With borrow SUBC	General register and data memory	SUBC r, m
			Data memory and immediate data	SUBC m, #n4

5.3.1 Addition and Subtraction When CMP = 0 and BCD = 0

Addition and subtraction are performed in 4-bit binary and the result is stored in the general register or data memory.

When the result of the operation is greater than 1111B (carry generated) or less than 0000B (borrow generated), the CY flag is set (1); otherwise it is reset (0).

When the result of the operation is 0000B, the Z flag is set (1) regardless of whether there is carry or borrow; otherwise it is reset (0).

5.3.2 Addition and Subtraction When CMP = 1 and BCD = 0

Addition and subtraction are performed in 4-bit binary.

However, because the CMP flag is set (1), the result of the operation is not stored in either the general register or data memory.

When there is a carry or borrow in the result of the operation, the CY flag is set (1); otherwise it is reset (0).

When the result of the operation is 0000B, the previous state of the Z flag is maintained; otherwise it is reset (0).

5.3.3 Addition and Subtraction When CMP = 0 and BCD = 1

BCD operations are performed.

The result of the operation is stored in the general register or data memory. When the result of the operation is greater than 1001B (9D) or less than 0000B (0D), the carry flag is set (1), otherwise it is reset (0).

When the result of the operation is 0000B (0D), the Z flag is set (1), otherwise it is reset (0).

Operations in BCD are performed by first computing the result in binary and then by using the decimal conversion circuit to convert the result to decimal. For information concerning the binary to decimal conversion, see Table 5-2 in **Section 5.2.5**.

In order for operations in BCD to be performed properly, note the following:

(1) Result of an addition must be in the range 0D to 19D.

(2) Result of a subtraction must be in the range 0D to 9D, or in the range -10D to -1D.

The following shows which value is considered the CY flag in the range 0D to 19D (shown in hexadecimal):

$\underbrace{0}_{\text{CY}}$, 0000B to $\underbrace{1}_{\text{CY}}$, 0011B

The following shows which value is considered the CY flag in the range -10D to -1D (shown in hexadecimal):

$\underbrace{1}_{\text{CY}}$, 0110B to $\underbrace{1}_{\text{CY}}$, 1111B

When operations in BCD are performed outside of the limits of (1) and (2) stated above, the CY flag is set (1) and the result of operation is output as a value greater than or equal to 1010B (0AH).

5.3.4 Addition and Subtraction When CMP = 1 and BCD = 1

BCD operations are performed.

The result is not stored in either the general register or data memory.

In other words, the operations specified by CMP = 1 and BCD = 1 are both performed at the same time.

Example

```
MOV   RPL,  #0001B    ;  Sets the BCD flag (BCD = 1).
MOV   PSW, #1010B     ;  Sets the CMP and Z flag (CMP = 1, Z = 1) and resets the CY flag
                      ;  (CY = 0).
SUB   M1,   #0001B    ;  #
SUBC  M2,   #0010B    ;  $
SUBC  M3,   #0011B    ;  %
```

By executing the instructions in steps numbered #, $, and %, the twelve bits in memory locations M1, M2, and M3 and the immediate data (321) can be compared in decimal.

5.3.5 Warnings Concerning Use of Arithmetic Operations

When performing arithmetic operations with the program status word (PSWORD), caution should be taken with regard to the result of the operation being stored in the program status word.

Normally, the CY and Z flags in the program status word are set (1) or reset (0) according to the result of the arithmetic operation being executed. However, when an arithmetic operation is performed on the program status word itself, the result is stored in the program status word. This means that there is no way to determine if there is a carry or borrow in the result of the operation nor if the result of the operation is zero.

However, when the CMP flag is set (1), results of arithmetic operations are not stored. Therefore, even in the above case, the CY and Z flags will be properly set (1) or reset (0) according to the result of the operation.

5.4 LOGICAL OPERATIONS

As shown in Table 5-4, logical operations consist of logical OR, logical AND, and logical XOR. Accordingly, the logical operation instructions are OR, AND, and XOR.

The OR, AND, and XOR instructions can be performed on either the general register and data memory, or on data memory and immediate data. The operands of these instructions are specified in the same way as for arithmetic operations ("r, m" or "m, #n4").

Logical operations are not affected by the BCD or CMP flags in the program status word (PSWORD). The operations do not affect the CY and Z flags at all.

Table 5-4 Logical Operations

Logical operation	Logical OR	General register and data memory	OR r, m
		Data memory and immediate data	OR m, #n4
	Logical AND	General register and data memory	AND r, m
		Data memory and immediate data	AND m, #n4
	Logical XOR	General register and data memory	XOR r, m
		Data memory and immediate data	XOR m, #n4

Table 5-5 Table of True Values for Logical Operations

Logical AND C = A AND B			Logical OR C = A OR B			Logical XOR C = A XOR B		
A	B	C	A	B	C	A	B	C
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

5.5 BIT EVALUATIONS

As shown in Table 5-6, there are both TRUE (1) and FALSE (0) bit evaluation instructions.

The SKT instruction skips the next instruction when a bit is evaluated as TRUE (1) and the SKF instruction skips the next instruction when a bit is evaluated as FALSE (0).

The SKT and SKF instructions can only be used with data memory.

Bit evaluations are not affected by the BCD flag in the program status word (PSWORD). The evaluations do not affect the CY and Z flags at all. However, when an SKT or SKF instruction is executed, the CMP flag is reset (0).

Sections 5.5.1 and **5.5.2** explain TRUE (1) and FALSE (0) bit evaluations.

Table 5-6 Bit Evaluation Instructions

Bit evaluation	TRUE (1) bit evaluation SKT m, #n
	FALSE (0) bit evaluation SKF m, #n

5.5.1 TRUE (1) Bit Evaluation

The TRUE (1) bit evaluation instruction (SKT m, #n) is used to determine whether or not the bits specified by n in the four bits of data memory m are TRUE (1). When all bits specified by n are TRUE (1), this instruction causes the next instruction to be skipped.

Example

```
MOV     M1,     #1011B
SKT     M1,     #1011B    ;  #
BR      A
BR      B
SKT     M1,     #1101B    ;  $
BR      C
BR      D
```

In this example, bits b_3, b_1, and b_0 of data memory M1 are evaluated in step number #. Because all the bits are TRUE (1), the program branches to B. In step number $, bits b_3, b_2, and b_0 of data memory M1 are evaluated. Since b_2 of data memory M1 is FALSE (0), the program branches to C.

5.5.2 FALSE (0) Bit Evaluation

The FALSE (0) bit evaluation instruction (SKF m, #n) is used to determine whether or not the bits specified by n in the four bits of data memory m are FALSE (0). When all bits specified by n are FALSE (0), this instruction causes the next instruction to be skipped.

Example

```
MOV    M1,    #1001B
SKF    M1,    #0110B    ; #
BR     A                ;
BR     B                ;
SKF    M1,    #1110B    ; $
BR     C                ;
BR     D                ;
```

In this example, bits b_2 and b_1 of data memory M1 are evaluated in step number #. Because both bits are FALSE (0), the program branches to B. In step number $, bits b_3, b_2, and b_1 of data memory M1 are evaluated. Since b_3 of data memory M1 is TRUE (1), the program branches to C.

5.6 COMPARISON EVALUATIONS

As shown in Table 5-7, there are comparison evaluation instructions for determining if one value is "equal to", "not equal to", "greater than or equal to", or "less than" another.

The SKE instruction is used to determine if two values are equal. The SKNE instruction is used to determine two values are not equal. The SKGE instruction is used to determine if one value is greater than or equal to another and the SKLT instruction is used to determine if one value is less than another.

The SKE, SKNE, SKGE, and SKLT instructions perform comparisons between a value in data memory and immediate data. In order to compare values in the general register and data memory, a subtraction instruction is performed according to the values in the CMP and Z flags in the program status word (PSWORD). For more information concerning comparison of the general register and data memory, see **Section 5.3**.

Comparison evaluations are not affected by the BCD or CMP flags in the program status word (PSWORD). The evaluations do not affect the CY and Z flags at all.

Sections 5.6.1 to **5.6.4** explain the "equal", "not equal", "greater than or equal", and "less than" comparison evaluations.

Table 5-7 Comparison Evaluation Instructions

Comparison evaluation	Equal SKE m, #n4
	Not equal SKNE m, #n4
	Greater than or equal SKGE m, #n4
	Less than SKLT m, #n4

5.6.1 "Equal" Evaluation

The "equal" evaluation instruction (SKE m, #n4) is used to determine if immediate data and the contents of a location in data memory are equal.

This instruction causes the next instruction to be skipped when the immediate data and the contents of data memory are equal.

Example

```
MOV     M1,     #1010B
SKE     M1,     #1010B    ;  #
BR      A
BR      B
;
SKE     M1,     #1000B    ;  $
BR      C
BR      D
```

In this example, because the contents of data memory M1 and immediate data 1010B in step number # are equal, the program branches to B. In step number $, because the contents of data memory M1 and immediate data 1000B are not equal, the program branches to C.

5.6.2 "Not Equal" Evaluation

The "not equal" evaluation instruction (SKNE m, #n4) is used to determine if immediate data and the contents of a location in data memory are not equal.

This instruction causes the next instruction to be skipped when the immediate data and the contents of data memory are not equal.

Example

```
MOV     M1,     #1010B
SKNE    M1,     #1000B    ;  #
BR      A
BR      B
;
SKNE    M1,     #1010B    ;  $
BR      C
BR      D
```

In this example, because the contents of data memory M1 and immediate data 1000B in step number # are not equal, the program branches to B. In step number $, because the contents of data memory M1 and immediate data 1010B are equal, the program branches to C.

5.6.3 "Greater Than or Equal" Evaluation

The "greater than or equal" evaluation instruction (SKGE m, #n4) is used to determine if the contents of a location in data memory is a value greater than or equal to the value of the immediate data operand. If the value in data memory is greater than or equal to that of the immediate data, this instruction causes the next instruction to be skipped.

Example

```
MOV     M1,     #1000B
SKGE    M1,     #0111B    ; #
BR      A
BR      B
;
SKGE    M1,     #1000B    ; $
BR      C
BR      D
;
SKGE    M1,     #1001B    ; %
BR      E
BR      F
```

In this example, the program will first branch to B since the value in data memory is larger than that of the immediate data (#). Next it will branch to D since the value in data memory is equal to that of the immediate data ($). Last it will branch to E since the value in data memory is less than that of the immediate data (%).

5.6.4 "Less Than" Evaluation

The "less than" evaluation instruction (SKLT m, #n4) is used to determine if the contents of a location in data memory is a value less than that of the immediate data operand. If the value in data memory is less than that of the immediate data, this instruction causes the next instruction to be skipped.

Example

```
MOV     M1,     #1000B
SKLT    M1,     #1001B    ; #
BR      A
BR      B
;
SKLT    M1,     #1000B    ; $
BR      C
BR      D
;
SKLT    M1,     #0111B    ; %
BR      E
BR      F
```

In this example, the program will first branch to B since the value in data memory is less than that of the immediate data (#). Next it will branch to C since the value in data memory is equal to that of the immediate data ($). Last it will branch to E since the value in data memory is greater than that of the immediate data (%).

5.7 ROTATIONS

There are rotation instructions for rotation to the right and for rotation to the left.

The RORC instruction is used for rotation to the right.

The RORC instruction can only be used with the general register.

Rotation using the RORC instruction is not affected by the BCD or CMP flags in the program status word (PSWORD). The rotation does not affect the Z flag at all.

Rotation to the left is performed by using the addition instruction ADDC.

Sections 5.7.1 and **5.7.2** explain rotation.

5.7.1 Rotation to the Right

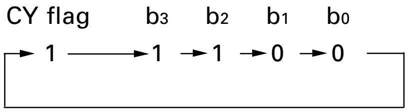
The instruction used for rotation to the right (RORC r) rotates the contents of the general register in the direction of its least significant bit.

When this instruction is executed, the contents of the CY flag becomes the most significant bit of the general register (bit b_3) and the least significant bit of the general register (bit b_0) is placed in the CY flag.

Examples 1.

```
MOV    PSW, #0100B    ;  Sets CY flag to 1.
MOV    R1,   #1001B
RORC   R1             ;  #
```

When these instructions are executed, the following operation is performed.



Basically, when rotation to the right is performed, the following operation is executed:

CY flag → b_3, b_3 → b_2, b_2 → b_1, b_1 → b_0, b_0 → CY flag.

2.

```
MOV    PSW, #0000B    ;  Resets CY flag to 0.
MOV    R1,   #1000B
MOV    R2,   #0100B
MOV    R3,   #0010B
RORC   R1
RORC   R2
RORC   R3
```

The program code above rotates the twelve bits in R1, R2, and R3 to the right.

5.7.2 Rotation to the Left

Rotation to the left is performed by using the addition instruction, "ADDC r, m".

Example

```
MOV     PSW,   #0000B    ; Resets CY flag to 0.
MOV     R1,    #1000B
MOV     R2,    #0100B
MOV     R3,    #0010B
ADDC    R3, R3
ADDC    R2, R2
ADDC    R1, R1
```

The program code above rotates the twelve bits in R1, R2, and R3 to the left.

6. PORTS

6.1 PORT 0B ($P0B_0/RLS_{HALT}$, $P0B_1/RLS_{STOP}$, $P0B_2$)

Port 0B is a three-bit input/output port. Only N-ch open-drain outputs appear on the pins of port 0B. The N-ch open-drain output mode allows application of 9 V, so it can be used for interfacing with a circuit operating on a different power supply voltage.

Input and output are set in units of nibbles. The input mode is set at reset, and the output mode is set by writing data to the port register in address 71H of the data memory. The output mode is maintained until the system is reset.

Output to the port is executed via the port register. Once data is written to the port register, all pins of port 0B are placed in the output mode to continue to output written data. The data is retained unless new data is written to the register.

Writing 1 to the port register makes the N-ch open-drain output pin high-impedance. Therefore, the pin which outputs 1 can be used as an input pin.

Whenever the port register is read, the read data indicates the states of the pins **Note**, not the contents of the port register, regardless of whether the pins are in the input or output mode. In this case, the contents of the port register remain unchanged.

The port register for port 0B consists of four bits but its highest bit is always set to 0. This means that if an attempt is made to write data to the highest bit of 71H, the data is invalidated and if an attempt is made to read it, 0 is always returned.

A $P0B_0$ input signal releases the HALT mode as a pseudo interrupt. A $P0B_1$ input signal releases the STOP mode as a pseudo interrupt. (See **Chapter 7**.)

6.2 PORT 0C ($P0C_0$ TO $P0C_3$)

Port 0C is a four-bit input/output port. CMOS (push-pull) outputs appear on those pins.

Input and output are set in units of nibbles. The input mode is set at reset, and the output mode is set by writing data to the port register in address 72H of the data memory. The output mode is maintained until the system is reset.

Output to the port is executed via the port register. Once data is written to the port register, all pins of the port 0C are placed in the output mode to continue to output written data. The data is retained unless new data is written to the register.

Whenever the port register is read, the read data indicates the states of the pins **Note**, not the contents of the port register, regardless of whether the pins are in the input or output mode. In this case, the contents of the port register remain unchanged.

6.3 PORT 0D ($P0D_0$ TO $P0D_3$)

Port 0D is a four-bit input/output port. CMOS (push-pull) outputs appear on these pins.

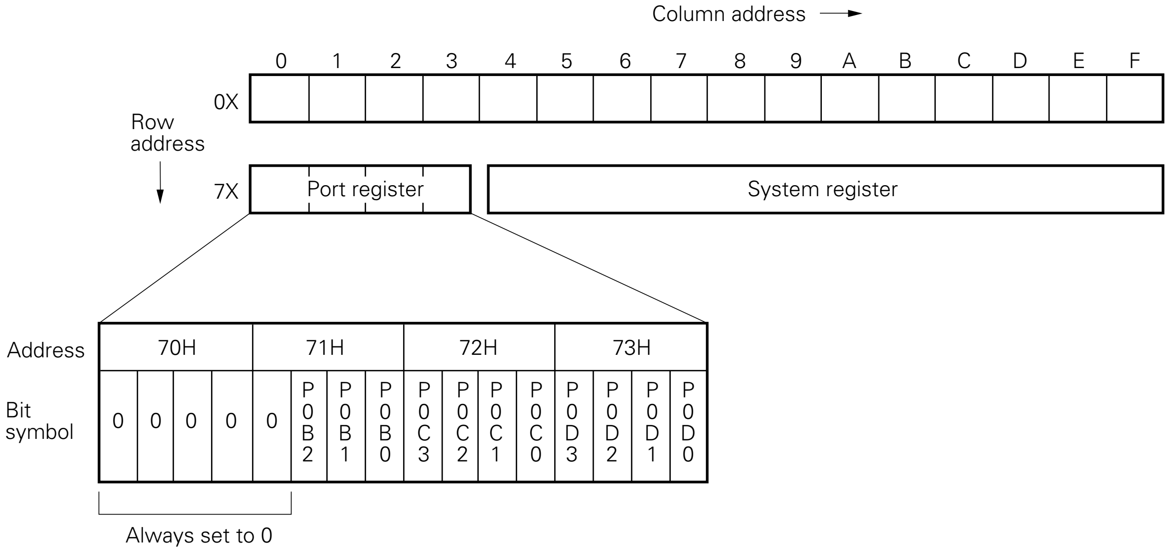
Input and output are set in units of nibbles. The input mode is set at reset, and the output mode is set by writing data to the port register in address 73H of the data memory. The output mode is maintained until the system is reset.

Output to the port is executed via the port register. Once data is written to the port register, all pins of the port 0D are placed in the output mode to continue to output written data. The data is retained until new data is written to the register.

Whenever the port register is read, the read data indicates the states of the pins **Note**, not the contents of the port register, regardless of whether the pins are in the input or output mode. In this case, the contents of the port register remain unchanged.

Note In the output mode, design an external circuit appropriately depending on the output data.

Fig. 6-1 Port Register Map



★ 6.4 NOTES ON MANIPULATING PORT REGISTERS

The states of the I/O port pins of the μPD17107 can be read even when the port pins have been set to output mode.

When a port register is manipulated with a built-in macro instruction (such as SETn or CLRn) or an AND, OR, or XOR instruction, the states of those pins for which the state should remain unchanged may change unexpectedly.

Especially when using some of the port 0B pins (N-ch open-drain outputs) as input pins, with the remaining port 0B pins being used as output pins, always take the possibility of this change in the states of the pins into consideration.

When a CLR1 P0B2 instruction (identical to an AND 71H, #1011B instruction) is applied to the port 0B pins, the corresponding port register and internal states are changed, as shown in Fig. 6-2.

Assume that the states of port 0B are those shown in Fig. 6-2 #. Pins P0B$_3$ and P0B$_2$, used as output pins, output high level, while pins P0B$_1$ and P0B$_0$, used as input pins, receive low level.

(It is required that high level be output, inside the chip, from the port 0B pins to be used as input pins. Although the μPD17103, μPD17103L, μPD17107, and μPD17107L do not support pin P0B$_3$, it is virtually assumed to exist within a program.)

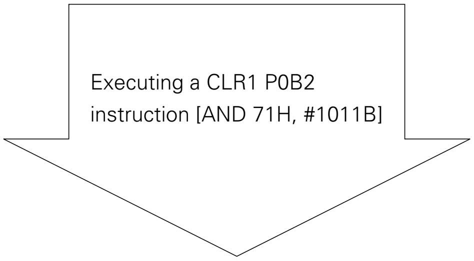
When a CLR1 P0B2 instruction is executed to set pin P0B$_2$ to low, the states of the port 0B pins change as shown in Fig. 6-2 $. The port register changes such that pins P0B$_1$ and P0B$_0$, required to output high level, actually output low level. This is because the CLR1 P0B2 instruction has been applied to the states of the port 0B pins, but not to the states of the port register.

To prevent this problem, use another instruction, such as a MOV instruction, to specify the states of all port 0B pins, not merely the states of those pins whose states are to be changed. In this example, it is recommended that a MOV 71H, #1011B instruction be used to set only pin P0B$_2$ to low.

Fig. 6-2 Changes in the Port Register According to the Execution of a CLR1 P0B2 Instruction

 # Before the instruction is executed

State	P0B$_3$	P0B$_2$	P0B$_1$	P0B$_0$
Port register	1	1	1	1
Internal	H output	H output	H output	H output
Pin	H	H	L (input)	L (input)



$ After the instruction is executed

State	P0B$_3$	P0B$_2$	P0B$_1$	P0B$_0$
Port register	1	0	0	0
Internal	H output	L output	L output	L output
Pin	H	L	L	L

H: High level, L: Low level

7. STANDBY FUNCTIONS

The μPD17107 provides two standby modes, the HALT mode and the STOP mode.

7.1 HALT MODE

The HALT mode stops the program counter (PC) while allowing the system clock to continue operating. The HALT mode can be entered with the HALT instruction, and can be released by a reset signal ($\overline{\text{RESET}}$) or high-level input to the P0B$_0$ pin. When the HALT mode is released by a high-level signal input to the P0B$_0$ pin, the system does not wait for the system clock oscillation to settle. The instruction immediately after the HALT instruction is executed.

When the HALT mode is released forcibly by the reset signal ($\overline{\text{RESET}}$), normal reset occurs, and the program starts at address 0H.

7.2 STOP MODE

The STOP mode stops the system clock oscillation so that data can be retained at low power voltage. The STOP mode can be entered with the STOP instruction, and can be released by a reset signal ($\overline{\text{RESET}}$) or high-level input to the P0B$_1$ pin. When the mode is released by a high-level signal input to the P0B$_1$ pin, the program starts with the instruction immediately after the STOP instruction.

When the STOP mode is released forcibly by the reset signal ($\overline{\text{RESET}}$), normal reset occurs, and the program starts at address 0H.

7.3 SETTING AND RELEASING THE STANDBY MODES

(1) Setting and releasing the HALT mode

Conditions for releasing the HALT mode are selected with the least significant bit of the operand in the HALT instruction as shown in Table 7-1. The high-order three bits of the operand must be set to 0.

Table 7-1 Conditions for Setting/Releasing the HALT Mode ★

HALT 000$\underline{\text{X}}$B ← 4-bit data in the operand

X	Conditions for setting/releasing the HALT mode
0	After executing a HALT instruction, the system enters the HALT mode unconditionally. The mode can be released only by the reset signal ($\overline{\text{RESET}}$). After the mode is released, the program starts at address 0H.
1	When a HALT instruction is executed with the P0B$_0$ pin being at low level, the system enters the HALT mode. The mode can be released by the reset signal ($\overline{\text{RESET}}$). When the mode is released, the program starts at address 0H. This mode can also be released when a high-level signal is applied to the P0B$_0$ pin. In this case, the program starts with the instruction immediately after the HALT instruction. When a HALT instruction is executed with the P0B$_0$ pin being at high level, the instruction is ignored (regarded as a NOP instruction) and the system does not enter the HALT mode.

(2) Setting and releasing the STOP mode

Conditions to release the STOP mode are selected with the least significant bit of the operand in the STOP instruction as shown in Table 7-2. The high-order three bits of the operand must be set to 0.

★ **Table 7-2 Conditions for Setting/Releasing the STOP Mode**

STOP 000<u>X</u>B ← 4-bit data in the operand

X	Conditions for setting/releasing the STOP mode
0	After executing a STOP instruction, the system enters the STOP mode unconditionally. All peripheral circuits are placed in the same initial state as when the system is reset, then they stop operating. The mode can be released only by the reset signal ($\overline{\text{RESET}}$). After the mode is released, the program starts at address 0H.
1	When a STOP instruction is executed with the P0B$_1$ pin being at low level, the system enters the STOP mode. The mode can be released by the reset signal ($\overline{\text{RESET}}$). When the mode is released, the program starts at address 0H. This mode can also be released when a high-level signal is applied to the P0B$_1$ pin. In this case, the program starts with the instruction immediately after the STOP instruction. When a STOP instruction is executed with the P0B$_1$ pin being at high level, the instruction is ignored (regarded as a NOP instruction) and the system does not enter the STOP mode.

★ 7.4 HARDWARE STATUSES IN STANDBY MODE

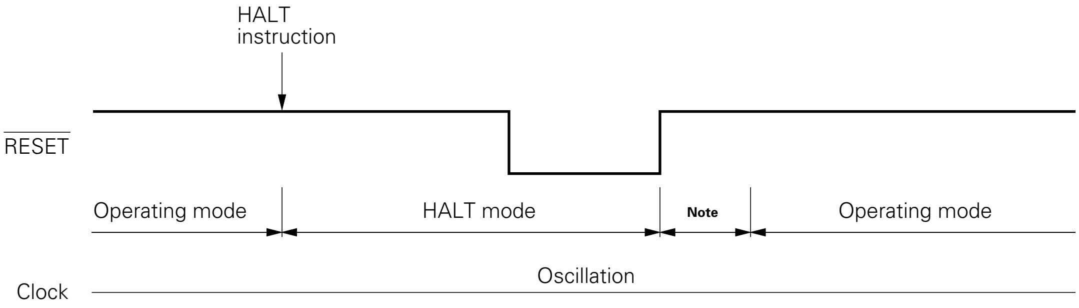
Hardware statuses in standby mode are as follows:

Table 7-3 Hardware Statuses in Standby Mode

Hardware	HALT or STOP 0001B instruction	STOP 0000B instruction
Clock generator	HALT instruction: Oscillation continued STOP instruction: Oscillation disabled	Oscillation disabled
Program counter	Address following a HALT or STOP instruction is indicated.	000H
Data memory (00H to 0FH)	Previous data is retained.	Previous data is retained.
Program status word (PSWORD)	Previous data is retained.	All bits are set to 0.
Port register (71H to 73H)	Previous data is retained. (Input/output mode of pins is also retained.)	Previous data is retained. (All pins are placed in input mode.)

7.5 TIMING FOR RELEASING THE STANDBY MODES

Fig. 7-1 Releasing the HALT Mode by $\overline{\text{RESET}}$ Input



When the $\overline{\text{RESET}}$ signal is applied to release the HALT mode, the $\overline{\text{RESET}}$ input makes a transition from low to high, then an operating mode is entered.

Note The HALT mode remains effective in this period, waiting for the operating mode. An operation starts after eight clock pulses on the OSC $_1$ pin are counted.

Fig. 7-2 Releasing the HALT Mode by High-Level Input to the P0B$_0$ Pin

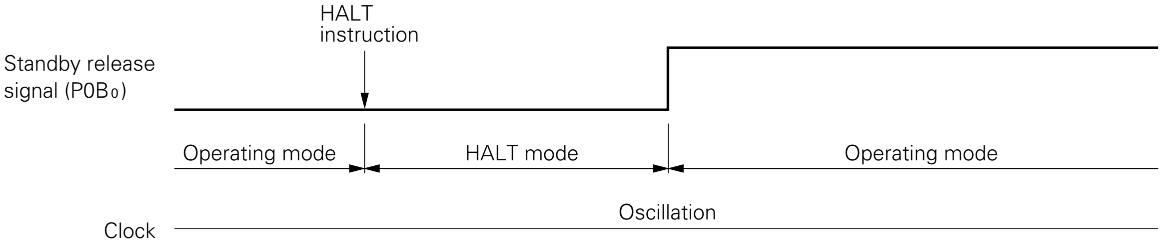
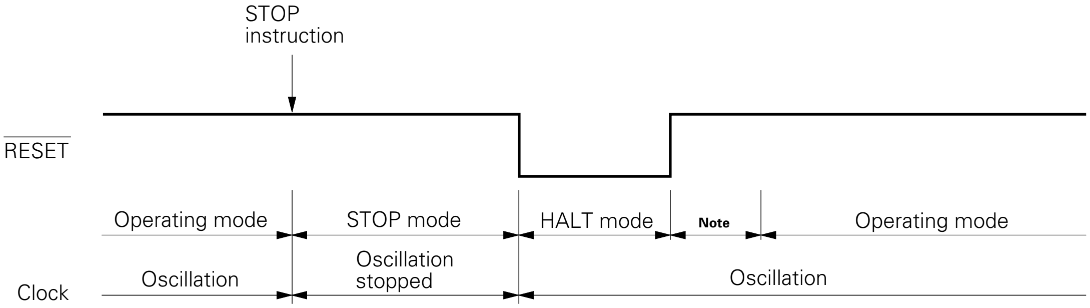


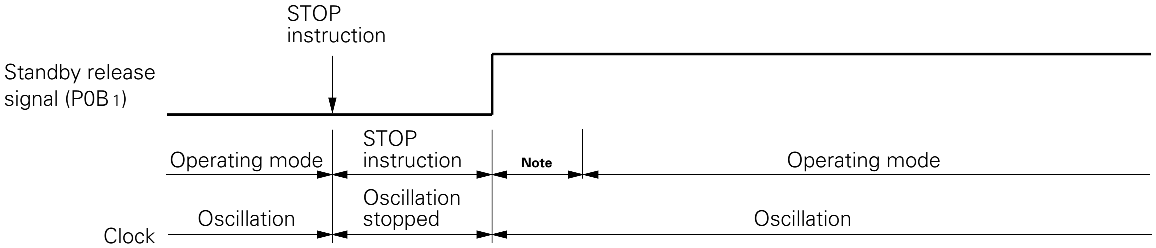
Fig. 7-3 Releasing the STOP Mode by $\overline{\text{RESET}}$ Input



As soon as the $\overline{\text{RESET}}$ input makes a transition from high to low in the STOP mode, the system clock starts generating clock pulses.

Note The HALT mode remains effective in this period, waiting for the generation of clock pulses to settle. An operation starts after eight clock pulses on the OSC $_1$ pin are counted.

Fig. 7-4 Releasing the STOP Mode by High-Level Input to the P0B$_1$ Pin



Note The HALT mode remains effective in this period, waiting for the generation of clock pulses to settle. An operation starts after eight clock pulses on the OSC$_1$ pin are counted.

8. RESET FUNCTION ★

8.1 RESET FUNCTION

A low-level signal, applied to the $\overline{\text{RESET}}$ pin, resets the system, then the hardware is initialized.

The system clock oscillates as long as the power supply voltage is supplied, even if a low-level signal is applied to the $\overline{\text{RESET}}$ pin.

A low to high transition on the $\overline{\text{RESET}}$ pin releases the reset status and causes the system to enter the operating mode once the 8-clock oscillation settling wait time has elapsed.

Table 8-1 Hardware Status after Reset

Hardware		• Reset immediately after power on • Reset during operation	Reset in standby mode[Note]
Program counter		000H	000H
Data memory (00H to 0FH)		Undefined	Data existing before reset is retained.
Program status word (PSWORD)		All bits are set to 0.	All bits are set to 0.
Port	Input/output mode	Input	Input
	Output latch	Undefined	Data existing before reset is retained.

Note The hardware is initialized when the STOP 0000B instruction is executed.

9. RESERVED WORDS USED IN ASSEMBLY LANGUAGE

9.1 MASK-OPTION PSEUDO INSTRUCTIONS

Source programs in the assembly language for the μPD17107 must include mask-option pseudo instructions to select pin options.

To do this, be sure to catalog the D17107.OPT file in AS17107 (device file for the μPD17107) into the current directory beforehand.

Specify mask options for the following pins:

- P0B$_0$
- P0B$_1$
- P0B$_2$
- $\overline{\text{RESET}}$

9.1.1 OPTION and ENDOP Pseudo Instructions

The part starting with the OPTION pseudo instruction and ending with the ENDOP pseudo instruction is referred to as a mask-option definition block. The coding format of the mask-option definition block is as follows.

Only the two pseudo instructions listed in Table 9-1 can be coded in the block.

Format:

Symbol	Mnemonic	Operand	Comment
[label:]	OPTION		[;comment]
	⋮		
	ENDOP		

9.1.2 Mask-Option Definition Pseudo Instructions

Table 9-1 lists the pseudo instructions to define a mask option for each pin.

Table 9-1 Mask-Option Definition Pseudo Instructions

Pin	Mask-option pseudo instruction	Number of operands	Operand
P0B$_2$ - P0B$_0$	OPTP0B	3	P0BPLUP (pull-up resistor provided) OPEN (no pull-up resistor provided)
$\overline{\text{RESET}}$	OPTRES	1	RESPLUP (pull-up resistor provided) OPEN (no pull-up resistor provided)

The coding format of OPTP0B is as follows. To define the mask option, specify P0B$_2$ (first operand), P0B$_1$, and P0B$_0$ in the operand field.

Format:

Symbol	Mnemonic	Operand	Comment
[label:]	OPTP0B	(P0B$_2$),(P0B$_1$),(P0B$_0$)	[;comment]

The coding format of OPTRES is as follows.

Format:

Symbol	Mnemonic	Operand	Comment
[label:]	OPTRES	($\overline{\text{RESET}}$)	[;comment]

Example The following mask options are set in a μPD17107 source file to be assembled:
P0B$_2$: Pull-up, P0B$_1$: Open, P0B$_0$: Open
$\overline{\text{RESET}}$: Pull-up

```
                              ⋮
;17107
 Setting mask options:   OPTION
                         OPTP0B      P0BPLUP, OPEN, OPEN
                         OPTRES      RESPLUP
                         ENDOP
                              ⋮
```

9.2 RESERVED SYMBOLS

Table 9-2 lists the reserved symbols defined in the μPD17107 device file (AS17107).

Table 9-2 Reserved Symbols

Name	Attribute	Value	R/W	Description
P0B0	FLG	0.71H.0	R/W	Bit 0 of port 0B
P0B1	FLG	0.71H.1	R/W	Bit 1 of port 0B
P0B2	FLG	0.71H.2	R/W	Bit 2 of port 0B
P0B3^{Note}	FLG	0.71H.3	R	Always set to 0
P0C0	FLG	0.72H.0	R/W	Bit 0 of port 0C
P0C1	FLG	0.72H.1	R/W	Bit 1 of port 0C
P0C2	FLG	0.72H.2	R/W	Bit 2 of port 0C
P0C3	FLG	0.72H.3	R/W	Bit 3 of port 0C
P0D0	FLG	0.73H.0	R/W	Bit 0 of port 0D
P0D1	FLG	0.73H.1	R/W	Bit 1 of port 0D
P0D2	FLG	0.73H.2	R/W	Bit 2 of port 0D
P0D3	FLG	0.73H.3	R/W	Bit 3 of port 0D
BCD	FLG	0.7EH.0	R/W	BCD arithmetic flag
PSW	MEM	0.7FH	R/W	Program status word
Z	FLG	0.7FH.1	R/W	Zero flag
CY	FLG	0.7FH.2	R/W	Carry flag
CMP	FLG	0.7FH.3	R/W	Compare flag

R/W: Read/write

Note Although a pin corresponding to P0B3 does not exist in the μPD17107, it is defined as a read-only flag so that it is treated as a dummy bit when a built-in macro is used.

10. INSTRUCTION SET

10.1 INSTRUCTION SET LIST

b_{15} b_{14}-b_{11}		0	1
BIN	HEX		
0 0 0 0	0	ADD r, m	ADD m, #n4
0 0 0 1	1	SUB r, m	SUB m, #n4
0 0 1 0	2	ADDC r, m	ADDC m, #n4
0 0 1 1	3	SUBC r, m	SUBC m, #n4
0 1 0 0	4	AND r, m	AND m, #n4
0 1 0 1	5	XOR r, m	XOR m, #n4
0 1 1 0	6	OR r, m	OR m, #n4
0 1 1 1	7	RET RETSK RORC r STOP s HALT h NOP	
1 0 0 0	8	LD r, m	ST m, r
1 0 0 1	9	SKE m, #n4	SKGE m, #n4
1 0 1 0	A		
1 0 1 1	B	SKNE m, #n4	SKLT m, #n4
1 1 0 0	C	BR addr	CALL addr
1 1 0 1	D		MOV m, #n4
1 1 1 0	E		SKT m, #n
1 1 1 1	F		SKF m, #n

10.2 INSTRUCTIONS

Legend

ASR : Address stack register pointed to by the stack pointer
addr: Program memory address (11 bits, high-order two bits are always set to 0)
CMP: Compare flag
CY : Carry flag
h : Halt release condition
m : Data memory address specified by m_R or m_C
m_R: Data memory row address (high order)
m_C: Data memory column address (low order)
n : Bit position (4 bits)
n4 : Immediate data (4 bits)
PC : Program counter
r : General register column address
SP : Stack pointer
s : Stop release condition
(×) : Contents addressed by ×

Instruction set	Mne-monic	Operand	Operation	Machine code			
				Op code	Operand		
Add	ADD	r,m	(r) ← (r) + (m)	00000	m_R	m_C	r
		m,#n4	(m) ← (m) + n4	10000	m_R	m_C	n4
	ADDC	r,m	(r) ← (r) + (m) + CY	00010	m_R	m_C	r
		m,#n4	(m) ← (m) + n4 + CY	10010	m_R	m_C	n4
Subtract	SUB	r,m	(r) ← (r) - (m)	00001	m_R	m_C	r
		m,#n4	(m) ← (m) - n4	10001	m_R	m_C	n4
	SUBC	r,m	(r) ← (r) - (m) - CY	00011	m_R	m_C	r
		m,#n4	(m) ← (m) - n4 - CY	10011	m_R	m_C	n4
Logical operation	OR	r,m	(r) ← (r) ∨ (m)	00110	m_R	m_C	r
		m,#n4	(m) ← (m) ∨ n4	10110	m_R	m_C	n4
	AND	r,m	(r) ← (r) ∧ (m)	00100	m_R	m_C	r
		m,#n4	(m) ← (m) ∧ n4	10100	m_R	m_C	n4
	XOR	r,m	(r) ← (r) $\veebar$ (m)	00101	m_R	m_C	r
		m,#n4	(m) ← (m) $\veebar$ n4	10101	m_R	m_C	n4
Test	SKT	m,#n	CMP ← 0, if (m) ∧ n = n, then skip	11110	m_R	m_C	n
	SKF	m,#n	CMP ← 0, if (m) ∧ n = 0, then skip	11111	m_R	m_C	n
Compare	SKE	m,#n4	(m) - n4, skip if zero	01001	m_R	m_C	n4
	SKNE	m,#n4	(m) - n4, skip if not zero	01011	m_R	m_C	n4
	SKGE	m,#n4	(m) - n4, skip if not borrow	11001	m_R	m_C	n4
	SKLT	m,#n4	(m) - n4, skip if borrow	11011	m_R	m_C	n4
Rotation	RORC	r	→ CY → $(r)_{b3}$ → $(r)_{b2}$ → $(r)_{b1}$ → $(r)_{b0}$	00111	000	0111	r
Transfer	LD	r,m	(r) ← (m)	01000	m_R	m_C	r
	ST	m,r	(m) ← (r)	11000	m_R	m_C	r
	MOV	m,#n4	(m) ← n4	11101	m_R	m_C	n4
Branch	BR	addr	PC ← addr	01100	addr		
Subroutine	CALL	addr	SP ← SP - 1, ASR ← PC, PC ← addr	11100	addr		
	RET		PC ← ASR, SP ← SP + 1	00111	000	1110	0000
	RETSK		PC ← ASR, SP ← SP + 1 and skip	00111	001	1110	0000
Others	STOP	s	STOP	00111	010	1111	s
	HALT	h	HALT	00111	011	1111	h
	NOP		No operation	00111	100	1111	0000

10.3 ASSEMBLER (AS17K) BUILT-IN MACRO INSTRUCTIONS ★

Legend

flag n: FLG symbol

< > : Characters enclosed in < > can be omitted.

	Mnemonic	Operand	Operation	n
Built-in macro	SKTn	flag 1, ···flag n	if (flag 1) - (flag n) = all "1", then skip	1 ≤ n ≤ 4
	SKFn	flag 1, ···flag n	if (flag 1) - (flag n) = all "0", then skip	1 ≤ n ≤ 4
	SETn	flag 1, ···flag n	(flag 1) - (flag n) ← 1	1 ≤ n ≤ 4
	CLRn	flag 1, ···flag n	(flag 1) - (flag n) ← 0	1 ≤ n ≤ 4
	NOTn	flag 1, ···flag n	if (flag n) = "0", then (flag n) ← 1 if (flag n) = "1", then (flag n) ← 0	1 ≤ n ≤ 4
	INITFLG	<NOT> flag 1, ··· <<NOT> flag n>	if description = NOT flag n, then (flag n) ← 0 if description = flag n, then (flag n) ← 1	1 ≤ n ≤ 4

11. ELECTRICAL CHARACTERISTICS (COMMON TO THE μPD17107 AND μPD17107(A))

ABSOLUTE MAXIMUM RATINGS (T_A = 25 °C)

Parameter	Symbol	Conditions		Rated value	Unit
Supply voltage	V_{DD}			-0.3 to +7.0	V
Input voltage	V_I	P0C, P0D, $\overline{\text{RESET}}$		-0.3 to V_{DD} + 0.3	V
		P0B	When a built-in pull-up resistor is connected	-0.3 to V_{DD} + 0.3	V
			When a built-in pull-up resistor is not connected	-0.3 to +11	V
Output voltage	V_O	P0C, P0D		-0.3 to V_{DD} + 0.3	V
		P0B	When a built-in pull-up resistor is connected	-0.3 to V_{DD} + 0.3	V
			When a built-in pull-up resistor is not connected	-0.3 to +11	V
★ High-level output current	I_{OH}	Each of P0C and P0D		-5	mA
		Total of all pins		-15	mA
Low-level output current	I_{OL}	Each of P0B, P0C, and P0D		30	mA
		Total of all pins		100	mA
Operating ambient temperature	T_A			-40 to +85	°C
Storage temperature	T_{stg}			-65 to +150	°C
Allowable dissipation	P_d	T_A = 85 °C	16-pin plastic DIP	400	mW
			16-pin plastic SOP	190	

★ **Caution Absolute maximum ratings are rated values beyond which some physical damages may be caused to the product; if any of the parameters in the table above exceeds its rated value even for a moment, the quality of the product may deteriorate. Be sure to use the product within the rated values.**

CAPACITANCE (T_A = 25 °C, V_{DD} = 0 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Input capacitance	C_{IN}	f = 1 MHz 0 V for pins other than pins to be measured			15	pF
I/O capacitance	C_{IO}				15	pF

I/O: Input/output

DC CHARACTERISTICS (T_A = -40 to +85 °C, V_{DD} = 2.5 to 6.0 V)

Parameter	Symbol	Conditions		Min.	Typ.	Max.	Unit
High-level input voltage	V_{IH1}	P0C, P0D		$0.7V_{DD}$		V_{DD}	V
	V_{IH2}	$\overline{\text{RESET}}$		$0.8V_{DD}$		V_{DD}	V
	V_{IH3}	P0B	**Note 1**	$0.8V_{DD}$		V_{DD}	V
	V_{IH4}		**Note 2**	$0.8V_{DD}$		9	V
Low-level input voltage	V_{IL1}	P0C, P0D		0		$0.3V_{DD}$	V
	V_{IL2}	$\overline{\text{RESET}}$		0		$0.2V_{DD}$	V
	V_{IL3}	P0B		0		$0.2V_{DD}$	V
High-level output voltage	V_{OH}	P0C, P0D	V_{DD} = 4.5 to 6.0 V I_{OH} = -2 mA	V_{DD} – 2.0			V
			I_{OH} = -200 μA	V_{DD} – 1.0			V
Low-level output voltage	V_{OL}	P0B, P0C, P0D	V_{DD} = 4.5 to 6.0 V I_{OL} = 15 mA			2.0	V
			I_{OL} = 600 μA			0.5	V
High-level input leakage current	I_{LIH1}	P0C, P0D, $V_{IN} = V_{DD}$				5	μA
	I_{LIH2}	P0B, $V_{IN} = V_{DD}$				5	μA
	I_{LIH3}	P0B, V_{IN} = 9 V	**Note 2**			10	μA
Low-level input leakage current	I_{LIL1}	P0C, P0D, V_{IN} = 0 V				-5	μA
	I_{LIL2}	P0B, V_{IN} = 0 V				-5	μA
High-level output leakage current	I_{LOH1}	P0C, P0D, $V_{OUT} = V_{DD}$				5	μA
	I_{LOH2}	P0B, $V_{OUT} = V_{DD}$				5	μA
	I_{LOH3}	P0B, V_{OUT} = 9 V	**Note 2**			10	μA
Low-level output leakage current	I_{LOL}	P0B, P0C, P0D, V_{OUT} = 0 V				-5	μA
Built-in pull-up resistor for pin $\overline{\text{RESET}}$	R_{RES}			20	47	95	kΩ
Built-in pull-up resistor for pin P0B	R_{P0B}			5	15	30	kΩ
Power supply current**Note 3**	I_{DD1}	Operation mode	V_{DD} = 5 V ±10 % f_{CC} = 1.0 MHz ±20 %		0.4	1.2	mA
			V_{DD} = 3 V ±10 % f_{CC} = 250 kHz ±20 %		50	150	μA
	I_{DD2}	HALT mode	V_{DD} = 5 V ±10 % f_{CC} = 1.0 MHz ±20 %		0.3	0.9	mA
			V_{DD} = 3 V ±10 % f_{CC} = 250 kHz ±20 %		40	120	μA
	I_{DD3}	STOP mode	V_{DD} = 5 V ±10 %		0.1	10	μA
			V_{DD} = 3 V ±10 %		0.1	5	μA

Notes 1. When a built-in pull-up resistor is connected

2. When a built-in pull-up resistor is not connected

3. This current excludes the current which flows through the built-in pull-up resistors.

CHARACTERISTICS OF DATA MEMORY FOR HOLDING DATA ON LOW SUPPLY VOLTAGE IN THE STOP MODE

(T_A = -40 to +85 ˚C)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Data hold supply voltage	V_{DDDR}		2.0		6.0	V
Data hold supply current	I_{DDDR}	V_{DDDR} = 2.0 V		0.1	5.0	μA

AC CHARACTERISTICS (T_A = -40 to +85 ˚C, V_{DD} = 2.5 to 6.0 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
CPU clock cycle time (instruction execution time)	t_{CY}	V_{DD} = 4.5 to 6.0 V	6.6		160	μs
			26.6		160	μs
High/low level width on P0B$_0$ and P0B$_1$	t_{PBH} t_{PBL}		10			μs
High/low level width on $\overline{\text{RESET}}$	t_{RSH} t_{RSL}		10			μs

Remark $t_{CY} = 8/f_{CC}$ (f_{CC}: frequency of the system clock oscillator)

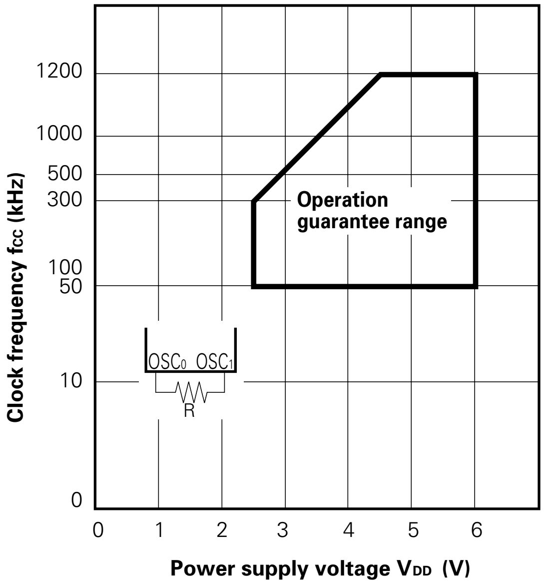
SYSTEM CLOCK OSCILLATOR CHARACTERISTICS (T_A = -40 to +85 ˚C)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
System clock oscillation frequency	f_{CC}	V_{DD} = 4.5 to 5.5 V, R_{OSC} = 24 kΩ	800	1000	1200	kHz
		V_{DD} = 2.7 to 3.3 V, R_{OSC} = 100 kΩ	200	250	300	kHz
		V_{DD} = 2.5 to 6.0 V, R_{OSC} = 100 kΩ	150	250	300	kHz

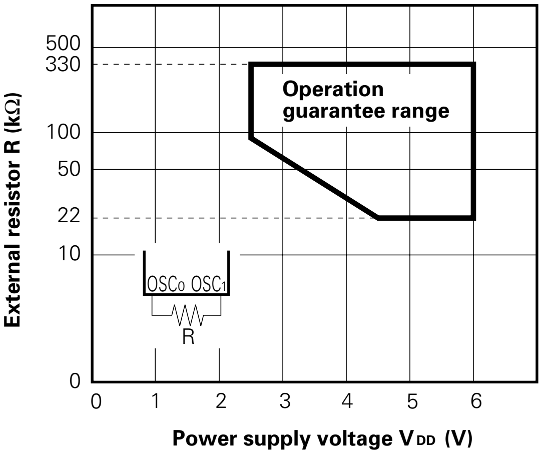
Caution The above conditions do not allow a resistance error.

12. CHARACTERISTIC CURVES (FOR REFERENCE)

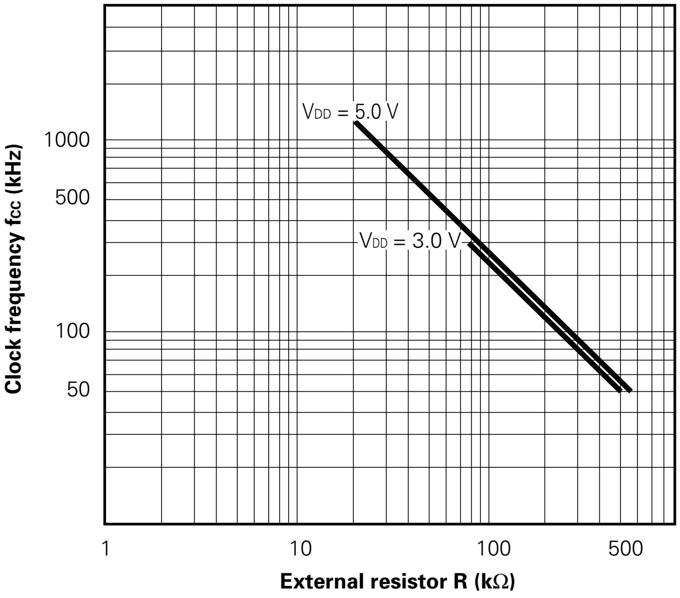
f_{CC} vs. V_{DD} for Operation Guarantee Range (T_A = -40 to +85 ˚C)



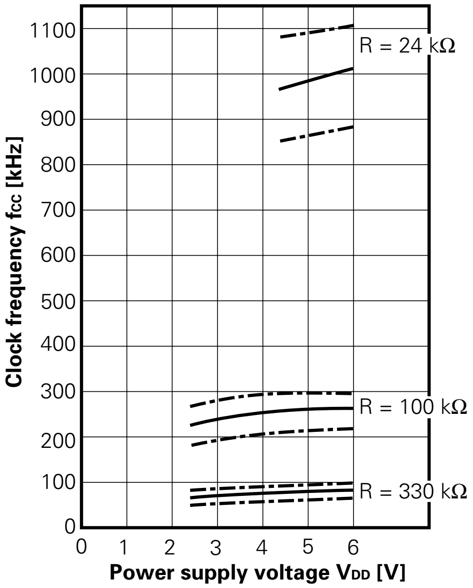
R vs. V_{DD} for Operation Guarantee Range (T_A = -40 to +85 ˚C)



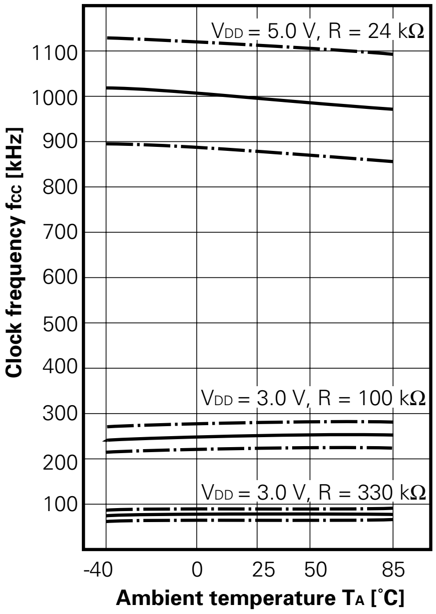
Typical Characteristics: f_{CC} vs. R (T_A = 25 ˚C)



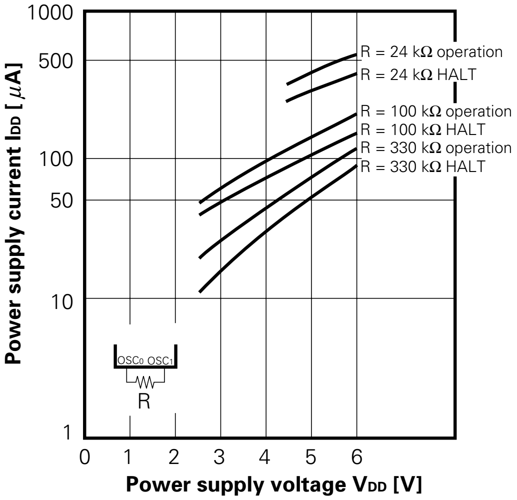
Typical Characteristics: f_{CC} vs. V_{DD} (T_A = -40 to +85 ˚C)



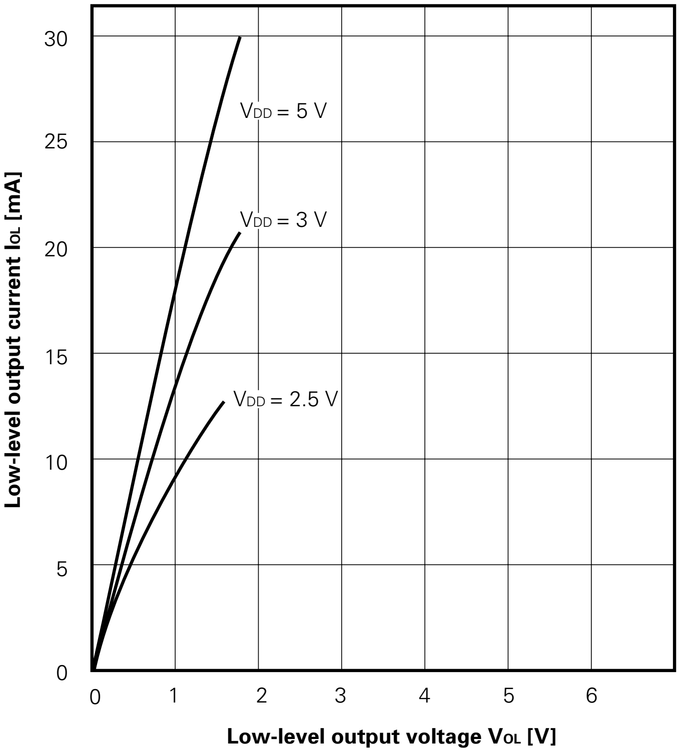
Typical Characteristics: f_{CC} vs. T_A



Typical Characteristics: I_{DD} vs. V_{DD} (T_A = 25˚C)

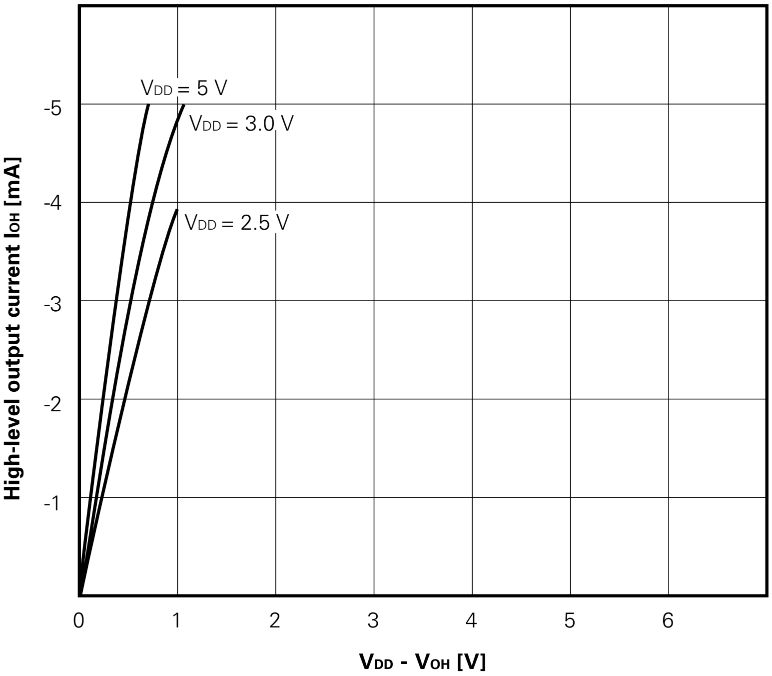


Typical Characteristics: I_{OL} vs. V_{OL} (T_A = 25 °C)



Caution **The maximum absolute rating is 30 mA per pin.**

Typical Characteristics: I_{OH} vs. (V_{DD} - V_{OH}) (T_A = 25 °C)

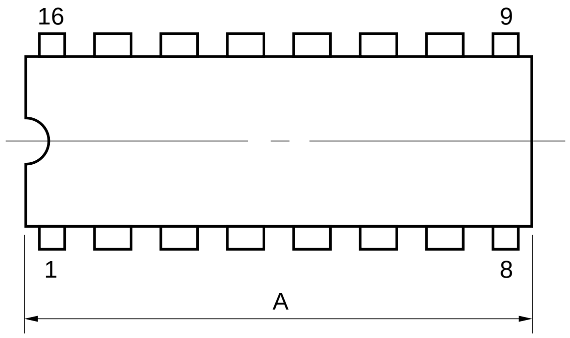


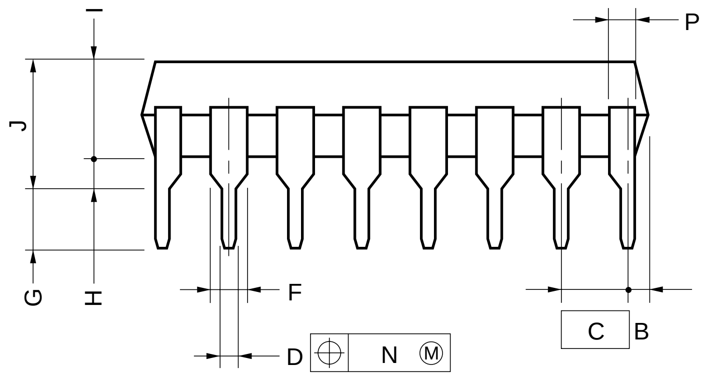
Caution **The maximum absolute rating is -5 mA per pin.**

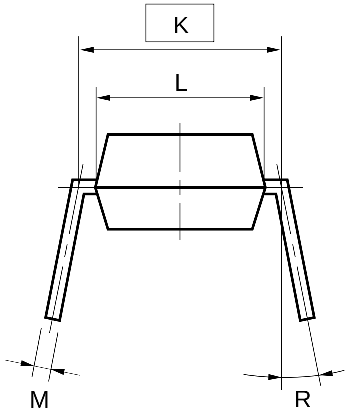
13. PACKAGE DRAWINGS

Package drawings of mass-produced products (1/2)

16 PIN PLASTIC DIP (300 mil)







NOTES

1) Each lead centerline is located within 0.25 mm (0.01 inch) of its true position (T.P.) at maximum material condition.

2) Item "K" to center of leads when formed parallel.

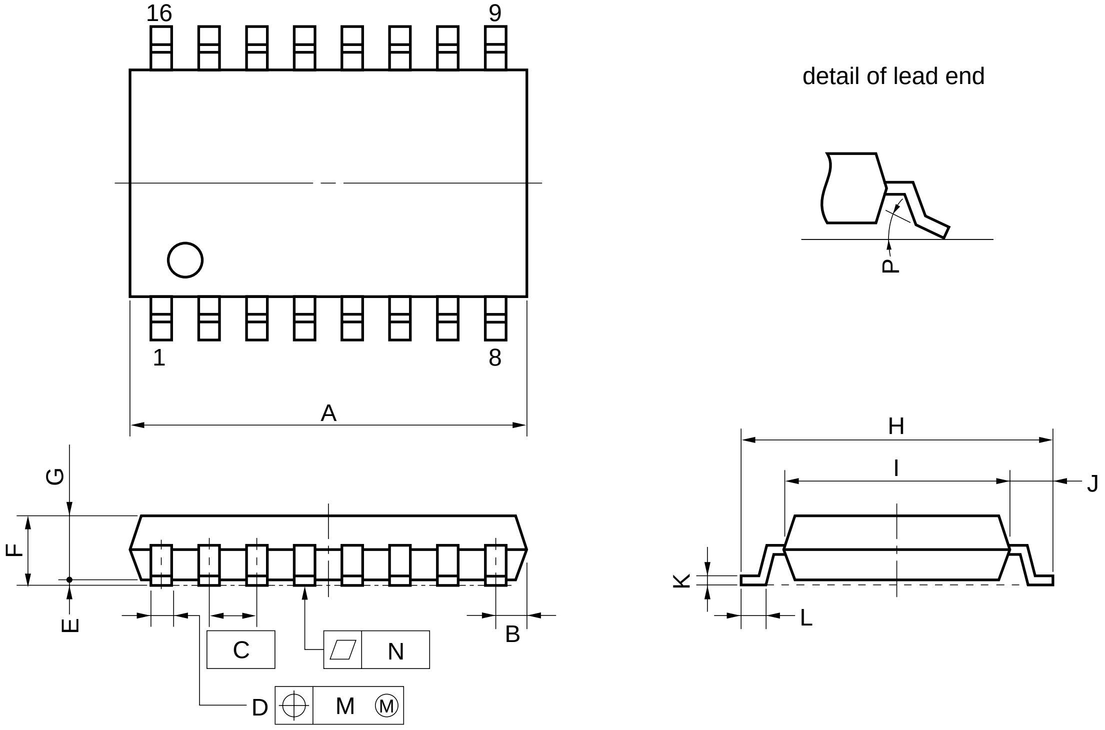
ITEM	MILLIMETERS	INCHES
A	20.32 MAX.	0.800 MAX.
B	1.27 MAX.	0.050 MAX.
C	2.54 (T.P.)	0.100 (T.P.)
D	0.50±0.10	$0.020^{+0.004}_{-0.005}$
F	1.1 MIN.	0.043 MIN.
G	3.5±0.3	0.138±0.012
H	0.51 MIN.	0.020 MIN.
I	4.31 MAX.	0.170 MAX.
J	5.08 MAX.	0.200 MAX.
K	7.62 (T.P.)	0.300 (T.P.)
L	6.5	0.256
M	$0.25^{+0.10}_{-0.05}$	$0.010^{+0.004}_{-0.003}$
N	0.25	0.01
P	1.1 MIN.	0.043 MIN.
R	0 ‘15°	0 ‘15°

P16C-100-300B-1

Caution The ES version is different from the corresponding mass-produced products in shape and material. See "ES package drawings (1/2)."

Package drawings of mass-produced products (2/2)

16 PIN PLASTIC SOP (300 mil)



NOTE

Each lead centerline is located within 0.12 mm (0.005 inch) of its true position (T.P.) at maximum material condition.

ITEM	MILLIMETERS	INCHES
A	10.46 MAX.	0.412 MAX.
B	0.78 MAX.	0.031 MAX.
C	1.27 (T.P.)	0.050 (T.P.)
D	$0.40^{+0.10}_{-0.05}$	$0.016^{+0.004}_{-0.003}$
E	0.1±0.1	0.004±0.004
F	1.8 MAX.	0.071 MAX.
G	1.55	0.061
H	7.7±0.3	0.303±0.012
I	5.6	0.220
J	1.1	0.043
K	$0.20^{+0.10}_{-0.05}$	$0.008^{+0.004}_{-0.002}$
L	0.6±0.2	$0.024^{+0.008}_{-0.009}$
M	0.12	0.005
N	0.10	0.004
P	$3^{\circ}{}^{+7^{\circ}}_{-3^{\circ}}$	$3^{\circ}{}^{+7^{\circ}}_{-3^{\circ}}$

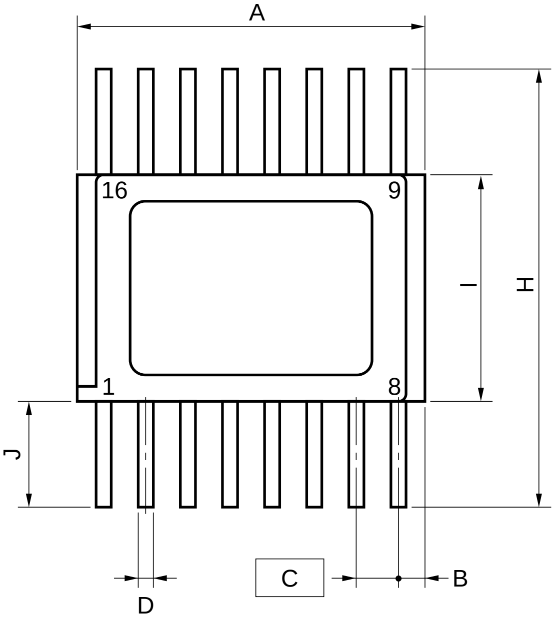
P16GM-50-300B-4

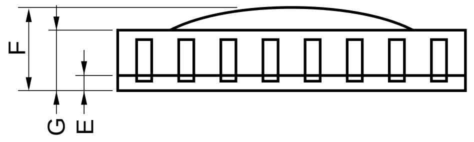
Caution **The ES version is different from the corresponding mass-produced products in shape and material. See "ES package drawings (2/2)."**

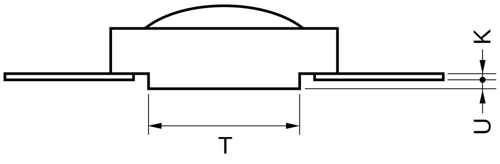
ES package drawings (1/2)

ES package drawings (2/2)

16 PIN CERAMIC SOP (FOR ES)







ITEM	MILLIMETERS	INCHES
A	10.16	0.40
B	1.02 MAX.	0.041 MAX.
C	1.27 (T.P.)	0.05 (T.P.)
D	0.43	0.017
E	0.48 MAX.	0.019 MAX.
F	2.54 MAX.	0.10 MAX.
G	1.7	0.067
H	12.1	0.476
I	6.9	0.272
J	2.35	0.093
K	0.13	0.005
T	4.8	0.189
U	0.025 MIN.	0.0 MIN.

X16B-50B-1

14. RECOMMENDED SOLDERING CONDITIONS

The conditions listed below shall be met when soldering the μPD17107.

For details of the recommended soldering conditions, refer to our document *SMD Surface Mount Technology Manual* (IEI-1207).

Please consult with our sales offices in case any other soldering process is used, or in case soldering is done under different conditions.

Table 14-1 Soldering Conditions for Surface-Mount Devices

μPD17107GS-×××　: 16-pin plastic SOP (300 mil)

★ **μPD17107GS(A)-××× : 16-pin plastic SOP (300 mil)**

Soldering process	Soldering conditions	Recommended conditions
Infrared ray reflow	Peak package's surface temperature: 230 ˚C Reflow time: 30 seconds or less (210 ˚C or more) Number of reflow processes: 1	IR30-00-1
VPS	Peak package's surface temperature: 215 ˚C Reflow time: 40 seconds or less (200 ˚C or more) Number of reflow processes: 1	VP15-00-1
★ Wave soldering	Solder temperature: 260 ˚C or less Flow time: 10 seconds or less Number of flow process: 1 Preheating temperature: 120 ˚C max. (measured on the package surface)	WS60-00-1
Partial heating method	Terminal temperature: 300 ˚C or less Heat time: 3 seconds or less (for one side of a device)	–

Caution Do not apply two or more different soldering methods to one chip (except for partial heating method for terminal sections).

Table 14-2 Soldering Conditions for Through Hole Mount Devices

μPD17107CX-×××　: 16-pin plastic DIP (300 mil)

★ **μPD17107CX(A)-××× : 16-pin plastic DIP (300 mil)**

Soldering process	Soldering conditions
Wave soldering (Only for terminals)	Solder temperature: 260 °C or less Flow time: 10 seconds or less
★ Partial heating method	Terminal temperature: 300 °C or less Heat time: 3 seconds or less (for each terminal)

Caution In wave soldering, apply solder only to the terminal. Care must be taken that jet solder does not come in contact with the main body of the package.

15. DIFFERENCES BETWEEN THE μPD17107, μPD17107L, and μPD17P107

Item	μPD17P107	μPD17107	μPD17107L
ROM	One-time PROM 1K bytes (512 × 16 bits)	Mask ROM 1K bytes (512 × 16 bits)	
Pull-up resistors of pins P0B$_0$-P0B$_2$	None	Mask option	
Pull-up resistor of $\overline{\text{RESET}}$ pin			
Connection pin	V_{PP} pin and operation mode selection pins are provided.	Neither V_{PP} pin nor operation mode selection pins are provided.	
Oscillation settling time (Counted in the number of clock pulses)	16	8	
Power supply voltage	2.5 to 6.0 V (at f_{CC} = 250 kHz) 4.5 to 6.0 V (at f_{CC} = 1 MHz)		1.5 to 3.6 V (at f_{CC} =200 kHz)
Oscillator characteristics[Note]	Differ depending on the type of microcontrollers (μPD17107, μPD17107L, or μPD17P107). See characteristic curves in the respective data sheets for details.		
Package	16-pin plastic DIP (300 mil) 16-pin plastic SOP (300 mil)		

Note When the supply voltage and the resistance of a resistor mounted externally are the same, the oscillation frequency of the μPD17P107 is about 10% lower than that of the μPD17107 or μPD17107L. Therefore, when the μPD17107 or μPD17107L is used instead of the μPD17P107, change the resistor externally mounted appropriately.

16. TINY MICROCONTROLLER FAMILY

Item	μPD17103	μPD17104	μPD17103L	μPD17104L	μPD17107	μPD17108	μPD17107L	μPD17108L
ROM capacity	1K bytes (512 × 16 bits)							
RAM capacity	16 × 4 bits							
Number of input/output port pins[Note]	11 (3)	16 (4)	11 (3)	16 (4)	11 (3)	16 (4)	11 (3)	16 (4)
System clock	Ceramic oscillation				RC oscillation			
Power supply voltage	2.7 to 6.0 V (at f_X = 2 MHz) 4.5 to 6.0 V (at f_X = 8 MHz)		1.8 to 3.6 V (at f_X = 2 MHz)		2.5 to 6.0 V (at f_{CC} = 250 kHz) 4.5 to 6.0 V (at f_{CC} = 1 MHz)		1.5 to 3.6 V (at f_{CC} = 200 kHz)	
Package	• 16-pin DIP • 16-pin SOP	• 22-pin shrink DIP • 24-pin SOP	• 16-pin DIP • 16-pin SOP	• 22-pin shrink DIP • 24-pin SOP	• 16-pin DIP • 16-pin SOP	• 22-pin shrink DIP • 24-pin SOP	• 16-pin DIP • 16-pin SOP	• 22-pin shrink DIP • 24-pin SOP
PROM version	μPD17P103	μPD17P104	μPD17P103	μPD17P104	μPD17P107	μPD17P108	μPD17P107	μPD17P108

Note A number in parentheses indicates the number of the N-ch open-drain outputs. N-ch open-drain outputs can be connected to internal pull-up resistors by specifying the mask option accordingly.

APPENDIX DEVELOPMENT TOOLS

The following support tools are available for developing programs for the μPD17107.

Hardware

Name	Description
In-circuit emulator IE-17K IE-17K-ET$^{\text{Note 1}}$ EMU-17K$^{\text{Note 2}}$	The IE-17K, IE-17K-ET, and EMU-17K are in-circuit emulators applicable to the 17K series. The IE-17K and IE-17K-ET are connected to the PC-9800 series (host machine) or IBM PC/AT™ through the RS-232-C interface. The EMU-17K is inserted into the extension slot of the PC-9800 series (host machine). Use the system evaluation board (SE board) corresponding to each product together with one of these in-circuit emulators. SIMPLEHOST™, a man machine interface, implements an advanced debug environment. The EMU-17K also enables user to check the contents of the data memory in real time.
SE board (SE-17107)	The SE-17107 is an SE board for the μPD17107, μPD17107L, or μPD17P107. It is used solely for evaluating the system. It is also used for debugging in combination with the in-circuit emulator.
Emulation probe (EP-17103CX)	The EP-17103CX is an emulation probe for the μPD17103, μPD17103L, μPD17P103, μPD17107, μPD17107L, or μPD17P107.
★ PROM programmer AF-9703$^{\text{Note 3}}$ AF-9704$^{\text{Note 3}}$ AF-9705$^{\text{Note 3}}$ AF-9706$^{\text{Note 3}}$	The AF-9703, AF-9704, AF-9705, and AF-9706 are PROM programmers for the μPD17P107. Use one of these PROM programmers with the program adapter, AF-9799, to write a program into the μPD17P107.
★ Program adapter (AF-9799$^{\text{Note 3}}$)	The AF-9799 is a socket unit for the μPD17P103, μPD17P104, μPD17P107, or μPD17P108. It is used with the AF-9703, AF-9704, AF-9705,or AF-9706.

Notes 1. Low-end model, operating on an external power supply

2. The EMU-17K is a product of IC Co., Ltd. Contact IC Co., Ltd. (Tokyo, 03-3447-3793) for details.

★ **3.** The AF-9703, AF-9704, AF-9705, AF-9706, and AF-9799 are products of Ando Electric Co., Ltd. Contact Ando Electric Co., Ltd. (Tokyo, 03-3733-1151) for details.

Software

Name	Description	Host machine	OS		Distribution media	Part number
17K series assembler (AS17K)	AS17K is an assembler applicable to the 17K series. In developing μPD17107 programs, AS17K is used in combination with a device file (AS17107).	PC-9800 series	MS-DOS™		5.25-inch, 2HD	μS5A10AS17K
					3.5-inch, 2HD	μS5A13AS17K
		IBM PC/AT	PC DOS™		5.25-inch, 2HC	μS7B10AS17K
					3.5-inch, 2HC	μS7B13AS17K ★
Device file (AS17107)	AS17107 is a device file for the μPD17107 and μPD17P107. It is used together with the assembler (AS17K) which is applicable to the 17K series.	PC-9800 series	MS-DOS		5.25-inch, 2HD	μS5A10AS17103 **Note**
					3.5-inch, 2HD	μS5A13AS17103 **Note**
		IBM PC/AT	PC DOS		5.25-inch, 2HC	μS7B10AS17103 **Note**
					3.5-inch, 2HC	μS7B13AS17103 **Note** ★
Support software (SIMPLEHOST)	SIMPLEHOST, running on the Windows™, provides man-machine-interface in developing programs by using a personal computer and the in-circuit emulator.	PC-9800 series	MS-DOS	Windows	5.25-inch, 2HD	μS5A10IE17K
					3.5-inch, 2HD	μS5A13IE17K
		IBM PC/AT	PC DOS		5.25-inch, 2HC	μS7B10IE17K
					3.5-inch, 2HC	μS7B13IE17K ★

Note μS××××AS17103 indicates the AS17103, AS17104, AS17107, AS17108, AS17103L, AS17104L, AS17107L, or AS17108L.

Remark The following table lists the versions of the operating systems described in the above table. ★

OS	Versions
MS-DOS	Ver. 3.30 to Ver. 5.00A**Note**
PC DOS	Ver. 3.1 to Ver. 5.0**Note**
Windows	Ver. 3.0 to Ver. 3.1

Note MS-DOS versions 5.00 and 5.00A and PC DOS Ver. 5.0 are provided with a task swap function. This function, however, cannot be used in these software packages.

[MEMO]

Cautions on CMOS Devices

Countermeasures against static electricity for all MOSs

Caution **When handling MOS devices, take care so that they are not electrostatically charged.** Strong static electricity may cause dielectric breakdown in gates. When transporting or storing MOS devices, use conductive trays, magazine cases, shock absorbers, or metal cases that NEC uses for packaging and shipping. Be sure to ground MOS devices during assembling. Do not allow MOS devices to stand on plastic plates or do not touch pins. Also handle boards on which MOS devices are mounted in the same way.

$ CMOS-specific handling of unused input pins

Caution **Hold CMOS devices at a fixed input level.**

Unlike bipolar or NMOS devices, if a CMOS device is operated with no input, an intermediate-level input may be caused by noise. This allows current to flow in the CMOS device, resulting in a malfunction. Use a pull-up or pull-down resistor to hold a fixed input level. Since unused pins may function as output pins at unexpected times, each unused pin should be separately connected to the V_{DD} or GND pin through a resistor.

If handling of unused pins is documented, follow the instructions in the document.

% Statuses of all MOS devices at initialization

Caution **The initial status of a MOS device is unpredictable when power is turned on.**

Since characteristics of a MOS device are determined by the amount of ions implanted in molecules, the initial status cannot be determined in the manufacture process. NEC has no responsibility for the output statuses of pins, input and output settings, and the contents of registers at power on. However, NEC assures operation after reset and items for mode setting if they are defined.

When you turn on a device having a reset function, be sure to reset the device first.

SIMPLEHOST is a trademark of NEC Corporation.
MS-DOS and Windows are trademarks of Microsoft Corporation.
IBM PC/AT and PC DOS are trademarks of IBM Corporation.

The export of this product from Japan is regulated by the Japanese government. To export this product may be prohibited without governmental license, the need for which must be judged by the customer. The export or re-export of this product from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.
NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others.
While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customer must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features.
NEC devices are classified into the following three quality grades:
"Standard", "Special", and "Specific". The Specific quality grade applies only to devices developed based on a customer designated "quality assurance program" for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers must check the quality grade of each device before using it in a particular application.

Standard: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots
Special: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)
Specific: Aircrafts, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

The quality grade of NEC devices in "Standard" unless otherwise specified in NEC's Data Sheets or Data Books.
If customers intend to use NEC devices for applications other than those specified for Standard quality grade, they should contact NEC Sales Representative in advance.
Anti-radioactive design is not implemented in this product.

M4 94.11