HITACHI DIGITAL
SIGNAL PROCESSOR(HSP) HD61810B

USER'S MANUAL

HITACHI

When using this manual, the reader should keep the following in mind:

1. This manual may, wholly or partially, be subject to change without notice.

2. All rights reserved: No one is permitted to reproduce or duplicate, in any form, the whole or part of this manual without Hitachi's permission.

3. Hitachi will not be responsible for any damage to the user that may result from accidents or any other reasons during operation of his unit according to this manual.

4. This manual neither ensures the enforcement of any industrial properties or other rights, nor sanctions the enforcement right thereof.

1. FEATURES

A High Performance Signal Processor HSP (Model HD61810B) aims at high speed digital signal processing and is composed of a single chip·based on a stored program format. Presented below are its features.

(1) Hardware

(i) Low power consumption, high speed VISI processor which avails itself of 3 μm CMOS LSI technology.

(ii) A dedicated multiplier and adder/subtractor are built in which permit high speed, high accuracy floating point operation.

(iii) High performance operation processing in implemented by pipe line control and horizontal micro instructions.

(iv) Large capacity memories are built in.
2-port accessible data RAM: 200 x 16 bits
Data ROM: 128 x 16 bits
Instruction ROM: 512 x 22 bits

(v) 8 bit and 16 bit microcomputers (6800, 68000) are compatible with an interface (synchronous interface).

(vi) A DMAC permits the exchange of DMA with external memories.

(vii) A serial I/O interface for up to 16 bits is provided.

(viii) Two levels of subroutine and interruption are provided.

(ix) An I/O transfer end interrupt function of one level and three factors is provided.

(x) Operating rates

Input clock: 16 MHz
Internal clock: 4 MHz
Instruction cycle: 250 ns
Product/Sum cycle (throughput by pipe line implementation): 250 ns

(xi) A single power supply of +5 V

(xii) Consumption power: 250 mW typ

(xiii) 40-pin ceramics/plastic package

(2) Software

(i) A horizontal type 22-bit microprogram is capable of reading instructions and data, multiplication and addition/subtraction, simultaneously, and is provided with high throughput instructions.

(ii) Multiplication and addition/subtraction are pipe line implemented, allowing high throughput product/ sum operations.

(iii) The address mode of data memories is adapted to signal processing operation.

(iv) Arithmetic and logical operation instructions are provided.

(v) Floating point and fixed point operations may be selected by instructions.

(vi) A repeat instruction permits high-speed repeated product/sum operations.

(vii) Data may be written into a control register through a microcomputer interface.

(vii) Addresses may be set externally to a program counter through the microcomputer interface.

2. SYSTEM CONFIGURATION

2.1 Terminal Functions

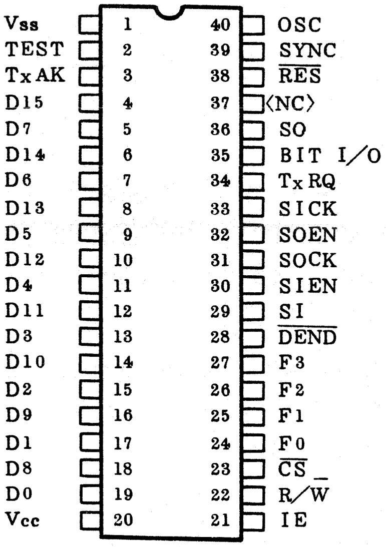


Fig. 2-1 Pin Arrangement (Top View)

Fig. 2-1 shows the HSP pin arrangement. The package is of a 40 pin dual line.

The following are the functions of each terminal:

(1) Vss, Vcc: Power supplies of 0V and 5 V.

(2) TEST: Input
Terminal for testing chips. (Fix the terminal to ground for use.)

(3) TxAK (Transfer Acknowledge): Input
To be used for a DMA transfer mode. DMA data transfer acknowledge input signal.

(4) D_{0-15} (Data Bus): Input/output, three states
Dual direction port having input and output registers. The direction is determined by a read/write (R/$\overline{W}$) control signal.

Unless a chip select ($\overline{CS}$) signal is active, the high-impedance state is established. 8-bit or 16-bit transfer mode may be selected by the contents set in an internal control register and those of function inputs (F_{0-3})

(5) IE (Interface Enable): Input
Data transfer timing signal of data buses (D_{0-15}). The data on a data bus are set to an internal register of the HSP by this signal, which becomes valid when $\overline{CS}$ is active.

An interruption may be produced inside the HSP using the fall of IE after a data transfer.

(6) R/$\overline{W}$ (Read/Write): Input
Control of switching the directions of data buses (D_{0-15}). The high level provides reading, while the low level writing. Valid, only when $\overline{CS}$ is active.

(7) $\overline{CS}$ (Chip Select): Input
A chip select input signal which makes a microcomputer interface valid. If this signal is put in the low level, D_{0-15}, F_{0-3}, R/$\overline{W}$ and IE become valid.

(8) F_{0-3} (Function Control): Input
Input of selectively controlling internal registers of the HSP during the exchange of data with the microcomputer. Connected to address buses of microcomputers (6800, 68000).

(9) $\overline{DEND}$ (DMA End): Input
DMA data transfer end signal. This signal, when it is put in the low level, terminates the DMA data transfer mode.

(10) SI (Serial data Input): Input
Enters serial input data into an internal shift register synchronously with a serial input clock (SICK). Data is entered into an internal shift register during the fall of SICK.

(11) SIEN (Serial Input Enable): Input

When this input goes high, serial input data commences being fetched into an internal shift register. At the completion of a fetch, an interruption may be produced in the HSP.

(12) SOCK (Serial Output Clock): Input

Serial data is output synchronously with this clock.

(13) SOEN (Serial Output Enable): Input

When this input is in the high level, serial output data is output from an internal shift register. An interruption may be produced in the HSP during the fall of this signal after the transfer of serial data.

When this signal is in the low level, the SO terminal goes high impedance.

(14) SICK (Serial Input Clock): Input

Serial data is entered synchronously with this clock.

(15) TxRQ (Transfer Request): Output, open drain

Used mainly in the DMA data transfer mode. This signal is used to request an external device (DMAC) to transfer the data stored in data buses D_{0-15}. In the DMA mode, this output is set upon the setting of the DMA transfer mode flag of a control register. Then, this output, until DEND is entered, automatically goes active (high), whenever data is exchanged with an accumulator, and is reset to the low level, after TxAK is entered. Also in the DMA transfer mode, data is transferred among I/O registers (IR, OR), accumulator (ACC) and memories in the HSP, through program control.

Merely used as an output terminal in the non-transfer DMA mode, permitting setting and resetting by the program.

(16) Bit I/O (Bit Input/Output): Input/output, open drain
One-bit bidirectional I/O terminal which may be set/ reset by the program. For input, the flip-flop for bit I/O in the HSP is to be set to "1". In that case, the input state of a terminal is read-in by an instruction.

(17) SO (Serial data Output): Output, three states
The serial output data of an internal shift register is output synchronously with the serial output clock (SOCK). Data is output from the rise of SOCK. When SOEN is in the low level, this output terminal goes high impedance.

(18) $\overline{\text{RES}}$ (Reset): Input
A reset terminal in the HSP. This input causes "0" to be set in a program counter and an interrupt mask flag I_M to be set, establishing an interrupt mask state. The other registers are not changed. When this input is released, the program starts. The program is executed from the address 1, and the instruction of the address 0 is not executed. (For resetting, the level "0" is to be entered for more than 1 μs with OSC applied).

(19) SYNC (Synchronous Clock): Input
Generates an internal operation clock of the HSP. Its frequency is 1/4 times as high as that of the input clock (OSC).

(20) OSC: Input
The fundamental clock used to operate the HSP. The standard frequency is 16 MHz.

2.2 Internal Functions

The HSP contains high-speed multipliers, adder/subtractor, programmable data ROMs, and instruction ROMs. The architecture is designed to be adapted to a wide range of applications. Hence, the HSP may be applied to voice processing and communication systems, etc. by writing coefficient and instruction sequence appropriate to an application system into data ROMs and instruction ROMs.

Fig. 2-2 shows the system block diagram of the HSP.

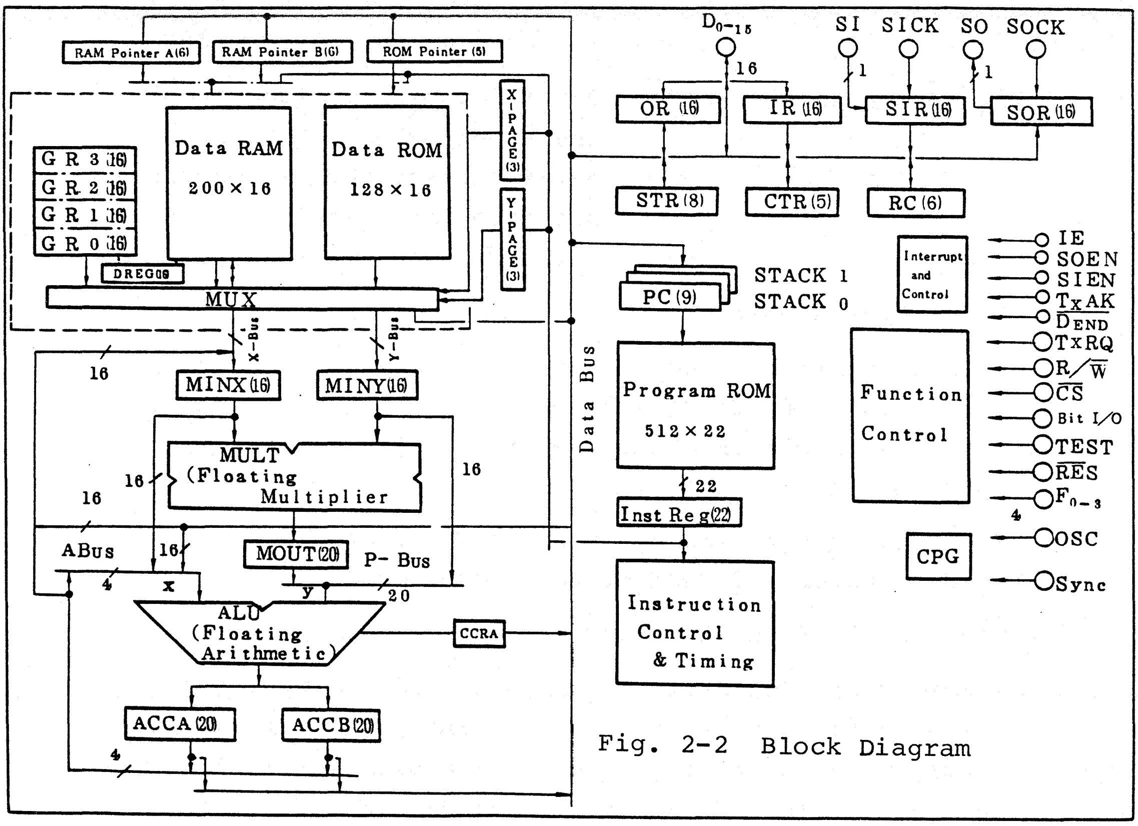


Fig. 2-2 Block Diagram

Presented below are the functions of components.

(1) IR (Input Register)

16-bit register. The data entered from data buses (D_{0-15}) are set at IE's timing.

The following two modes are provided to transfer data. The mode is selected using control flags in the control register (CTR).

(i) Word transfer mode: 16-bit data are set simultaneously. Interconnected to the 16-bit microcomputer (68000) in the sync interface mode.

(ii) Byte transfer mode: 8-bit data are divided into the upper bytes and lower bytes for setting. Compatible with the interface of the 8-bit microcomputer (6800).

(2) OR (Output Register)

16-bit register. The contents of this register are output to external data buses (D_{0-15}). Until data is set by an HSP's program, the previously set data is not changed and exists.

The data transfer mode, like the IR, falls into the word transfer mode and byte transfer mode.

(3) SIR (Serial Input Register)

16-bit shift register dedicated to serial input. The serial entry data from an SI terminal is input synchronously with the timing clock from a SICK terminal.
The entry value of the shift register is fed via internal 16-bit buses connected in parallel to an accumulator

(ACC) by an internal instruction. Up to 16 bits of serial input data may be entered.

The SIR is cleared, after data is transferred to an accumulator by the program.

(4) SOR (Serial Output Register)

16-bit shift register dedicated to serial output. The data set from an internal bus in the HSP in 16-bit parallel are output from a serial output terminal (SO) by bit synchronously with the serial output timing clock entered into the SOCK terminal. Up to 16 bits of output data may be selected, as the user desires. The number of output bits is determined by the number of shift clocks entered into the SOCK terminal during an active period of the input signal at a serial output enable terminal (SOEN).

The SOR is cleared, after data is output to the outside.

(5) Inst Reg (Instruction Register)

Register which buffers a 22-bit instruction read from an instruction ROM.

(6) PC (Program Counter)

A 9-bit address counter dedicated to the instruction ROM. The PC generates ROM addresses in the range from addresses 0 thru 511.

Addresses 0 thru 255 may be set by an external control via an external data bus. In that case, the run start instruction address is a set address plus 1.

(7) STACK0/1 (Stack Register)

9-bit stack register dedicated to save the PC. The contents of the PC are saved, when a subroutine jump or interruption occurs.

Two stack registers are provided, allowing nesting of two levels.

(8) RC (Repeat Counter)

6-bit down counter. Used to repeatedly execute the same instruction and to process the loop by a jump instruction.

The counter allows to reduce the program steps of repeated product/sum operation, and also the processing time period.

(9) ROM Pointer

5-bit address pointer dedicated to the data ROM. Generates the effective address of a data ROM by introducing the X/Y page address (3 bits) in an instruction code.

(10) RAM Pointer A/B

6-bit address pointer dedicated to a data RAM. Generates the effective address of a data RAM by introducing the X/Y page address (3 bits) in an instruction code. Two pointers having the same function are provided each of which may be selected by the instruction. Two RAM pointers allow complex operation programs such as FFT to be efficiently constructed.

(11) X/Y-Page (X/Y-Page Address Register)

3-bit buffer register dedicated to a page address. Generates the effective address of data ROM/RAM by combining its contents with the value of the ROM/RAM pointer.

(12) GR 0, 1, 2, 3 (General Register)

General 16-bit register. Used as a working register. The contents may be input/output only via a Y-bus.

(13) MINX (Multiplier Input X - Register)

Stores data from the X-bus or internal data bus, and holds it for multiplication. 16-bit register.

(14) MINY (Multiplier Input Y - Register)

Stores data from the Y-bus, and holds it for multiplication. 16-bit register.

(15) MOUT (Multiplier Output Register)

Buffer register which stores the output of a multiplier. Holds the output data of a multiplier for a one instruction cycle period. 20-bit register which is made up of a mantissa comprising 16 bits and an exponent part comprising 4 bits.

(16) DREG (Delay Register)

16-bit register. Holds data to be output to the Y-bus for one-instruction cycle period. Used to efficiently change the address which saves RAM's data. Validly used to shift one by one the data storage addresses on a data RAM for a one-sample delay function in signal processing.

(17) ACC A/B (Accumulator A/B)

20-bit accumulator. ALU's output is set. Two accumulators A/B are selected by the instruction.

(18) CCR (Condition Code Register)

Made up of condition flags which reflect the operation results of the ALU. Three flags are provided: carry (C), negative (N) and zero (Z).

(19) STR (Status Register)

8-bit register which reflects the inner conditions of the HSP.

(20) CTR (Control Register)

5-bit control register which sets conditions used to control the operation of the HSP. Able to be set by external control via an HSP's instruction and I/O terminals D_{0-15}.

(21) Data RAM

Its size is 200 words x 16 bits. Separated to four pages. Each page consists of 50 words. 2-word data may be read from a different page. Output to X/Y buses, and only one word is read in from the data bus.

(22) Data ROM

Its size is 128 words x 16 bits. Separated to four pages. Each page consists of 32 words. Only one word is read, and is output to the X or Y bus.

(23) Instruction ROM

Its size is 512 words x 22 bits. 22-bit instructions are simultaneously read to the Inst. Register during each instruction cycle.

Once reset, the instruction ROM starts from the address 0, whereas the instruction is executed from the address 1. If a jump to the address 0 occurs during execution of the instruction, the instruction stored in the address 0 is also normally executed.

Since the end address ($1FF) of the ROM is used as the vector address, the user must write a jump instruction dedicated to a jump to an interrupt processing routine. The user cannot use the address range from $1E7 to $1FE in the ROM, for they are used to save LSI test programs.

(24) MULT (Multiplier)

Devoted to floating point multiplication (mantissa 12 bits x 12 bits → 16 bits, exponent 4 bits + 4 bits → 4 bits) and fixed point multiplication (12 bits x 12 bits → 16 bits).

Each operation mode is switched by the instruction. Details will be described later.

(25) ALU (Arithmetic Logic Unit)

Devoted to arithmetic and logical operations. A floating point addition/subtraction mode (mantissa 16 bits, exponent 4 bits) or a fixed point addition/subtraction mode (16 bits) is selected by the instruction. Details will be described later.

3. ARITHMETIC OPERATION

3.1 General

The signal processing for voice and communication involves high-accuracy and high-speed arithmetic operations. The HSP embodies operation accuracy well adapted to signal processing, implementing a high-speed floating point operation circuit on a single-chip LSI. The floating point operation provides a dynamic range (maximum amplitude of operation data) necessary to improve operation accuracy in a smaller multiplier and data memory vis-à-vis the fixed point operation, and thus is advantageous in LSI implementation. The floating point operation is put in the spotlight as the architecture of the second generation for the signal processor.

The floating point operation of the HSP is designed to meet the accuracy required from the application system given by hatches, as shown in Fig. 3-1. This floating point operation avails itself of the feature that the significant bit length, like the fixed point operation, varies in proportion to the data amplitude, if is low, but is sufficiently met by 16 bits maximum, if the data amplitude exceeds 16 bits (2^{-8}). In accordance with the data amplitude, the fixed point operation method and the floating point operation method are automatically switched to each other.

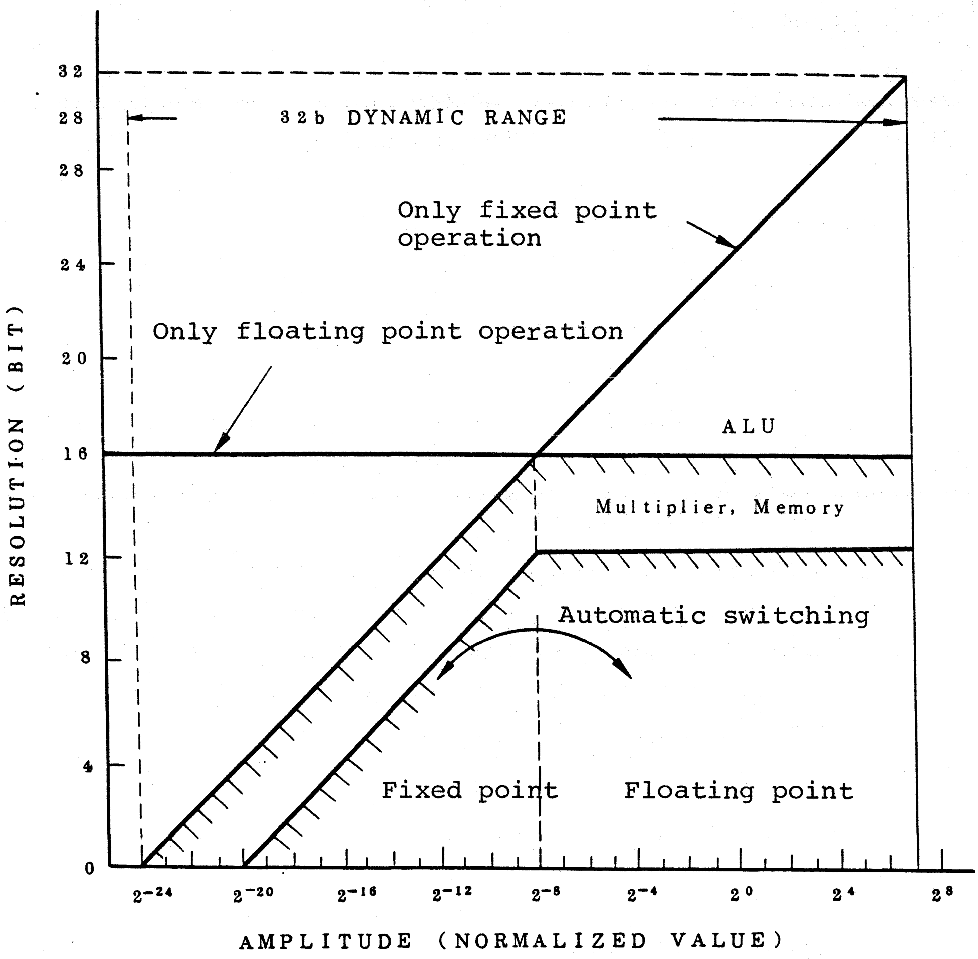


Fig. 3-1 Operation Accuracy of Signal Processing Application System

3.2 Data Format

The HSP has the data format shown in Fig. 3-2, applying to floating point and fixed point operations.

(1) Adder/Subtractor

(A) Fixed point data

(i) Binary expression: Expresses the range of 0 to $2^{16}-1$.

(ii) 2's complement expression : The most significant bit is a sign bit.

(B) Floating point data

Both mantissa and exponent part express data in 2's complement. For both the mantissa and exponent part, the most significant bit is a sign bit, and the decimal point is placed between the bit 2^{15} (most significant bit) and the bit 2^{14}.

Mantissa: Expresses the range of -1 to $1 - 2^{-15}$.

Exponent part: Expresses the range of -8 to 7.

(2) Multiplier

(A) Fixed point data

Expressed only in a 2's complement. The most significant bit is a sign bit. The input data is composed of 12 bits and the output data of 16 bits.

(B) Floating point data

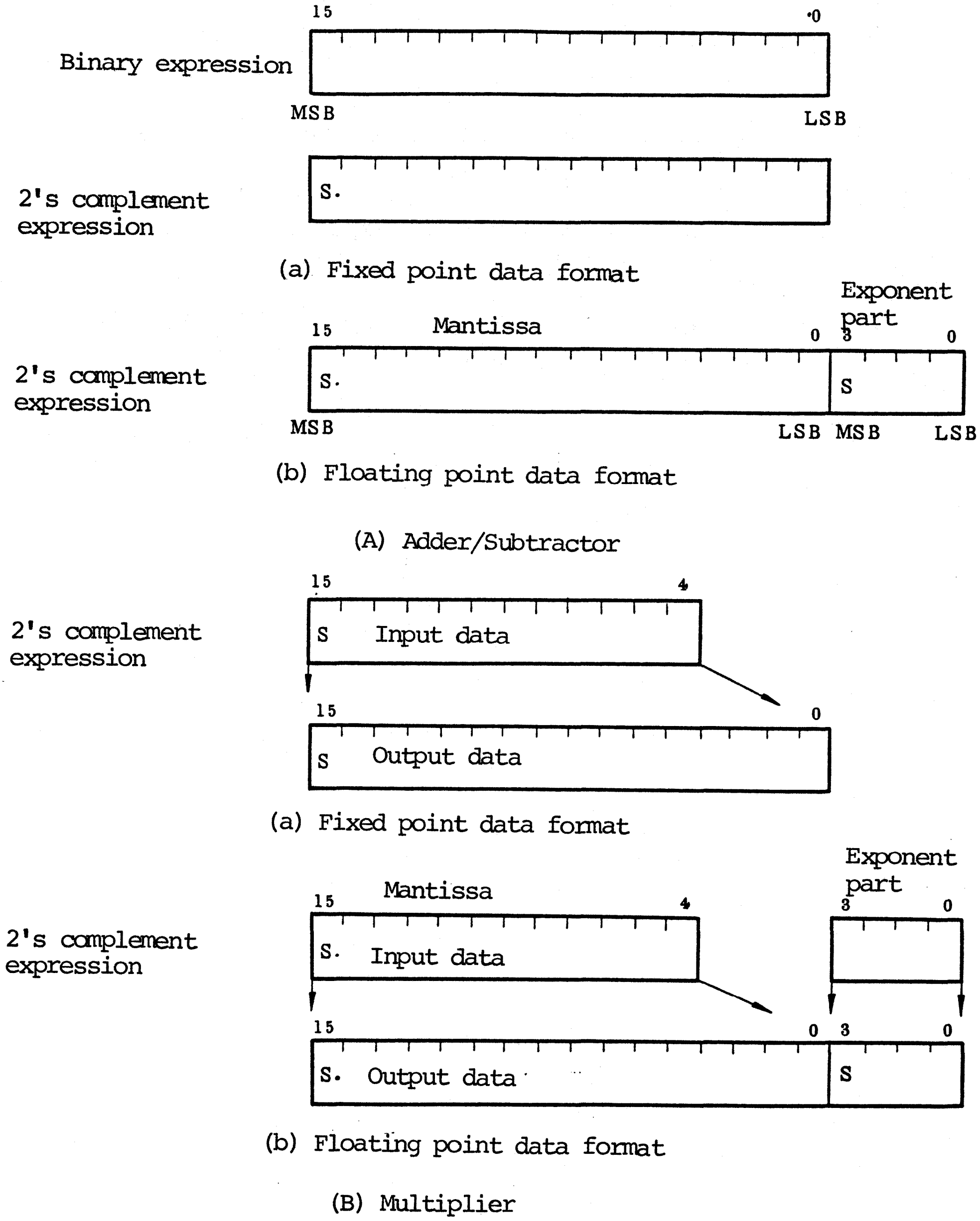
Both the mantissa and the exponent part are expressed only in 2's complements. The input data of the mantissa is composed of 12 bits, and the output data of 16 bits.

For the exponent part, both input and output data are composed of 4 bits. The decimal point is placed between the bit 2^{15} and 2^{14}.

Mantissa: Input: Capable of expressing the range of -1 to $1 - 2^{-11}$.

Output: Capable of expressing the range of -1 to $1 - 2^{-15}$.

Exponent part : Capable of expressing the range of -8 to 7.



(a) Fixed point data format

(b) Floating point data format

(A) Adder/Subtractor

(a) Fixed point data format

(b) Floating point data format

(B) Multiplier

Fig. 3-2 Data Format of Operational Circuit

3.3 Addition/Subtraction of Floating Point

(1) Arithmetic Operation

The HSP is provided with the dedicated floating point adder/subtractor circuit (FAUL) as shown in Fig. 3-3. Shifters for digit matching or normalization of floating point data are provided before and after the ALU which is engaged in fixed 16-bit point operation, implementing high-speed floating point operation. The arithmetic operation of a floating point is executed as shown in the following expression:

$$
\begin{aligned}
A = A_1 + A_2 &= a_1 \times 2^{C_1} + a_2 \times 2^{C_2} \\
&= 2^{C_1} \left(a_1 + a_2 \times 2^{(C_2 - C_1)} \right) && \text{; Digit matching} \\
&= a_3 \times 2^{C_1} \\
&= a \times 2^{C} && \text{; Normalization}
\end{aligned}
$$

Note) FALU input data : $A_1 = a_1 \times 2^{C_1}$, $A_2 = a_2 \times 2^{C_2}$

$(C_2 \leqq C_1)$

FALU output data : $A = a \times 2^{C}$

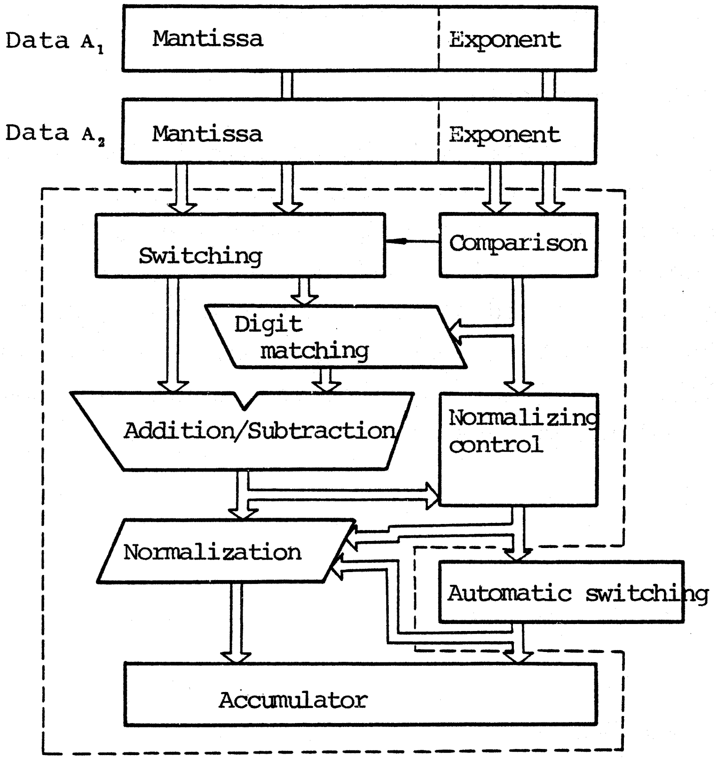


Fig. 3-3 Floating Point Adder/Subtractor Circuit

In the above operation, the value of A is transferred to the ACC, as it is, the exponent value C of the last output data is equal to or greater than -8 and is equal to or less than +7. If the value of C is out of the above range, the following overflow/underflow protection operations are carried out:

(A) Overflow

If C is greater than +7, the exponent part is fixed to +7 so that the absolute value of the mantissa may be made to be maximum (the sign is not changed). As a result, the output data is $(1 - 2^{-15}) \times 2^{7}$ or $(-1) \times 2^{7}$.

The protect operation is carried out for the overflow during the arithmetic operation of the fixed point too, when the overflow protection bit (OVEP) of the control register CTR is set ("1").

(B) Underflow

If C is less than -8, the dynamic range of arithmetic operation is extended using the above-mentioned floating point/fixed point switching method. In that case, no general normalization is carried out. The value of the exponent part is fixed to -8, while the bits of the mantissa are shifted to the left direction by (-8-C) in the normalizing shifter. Fig. 3-4 shows an example.

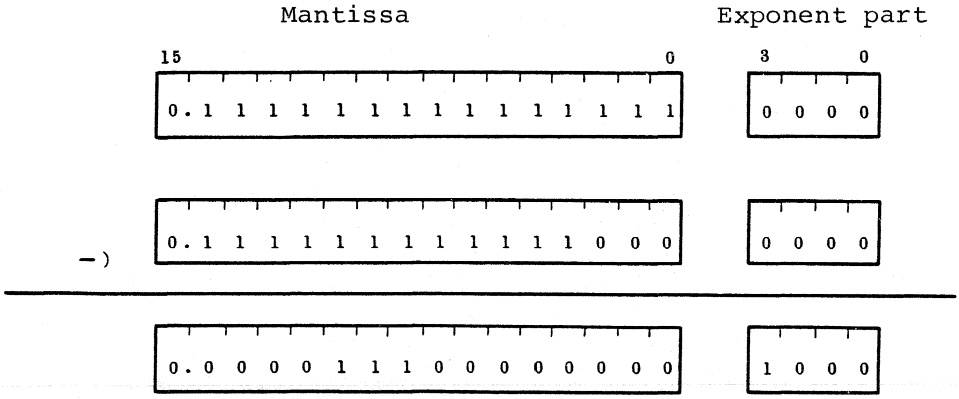


Fig. 3-4 Operation during Underflow

If the exponent parts of two inputs are ∸8, the above operation is conducted, as if arithmetic fixed point operation were carried out.

The above technique reduces the circuit scale of the whole LSI, implementing the data amplitude of 32-bit data.

(2) Operation Error of Floating Point Adder/Subtractor

(i) Truncation by digit matching

If there is a great difference between the values of the exponent parts of two floating point data inputs, the contents of the lower bit of the mantissa in data with the smaller exponent part are truncated during digit matching operation. In normalizing operation, "0" is carried from the lower bit, resulting in an error.

(ii) Error caused during subtraction

The floating point subtraction is essentially conducted as follows:

$$
\begin{aligned}
A_1 - A_2 &= N_1 \cdot 2^{M_1} - N_2 \cdot 2^{M_2} \\
&= N_1 \cdot 2^{M_1} + (\overline{N_2} + 1) \cdot 2^{M_2} \\
&= \begin{cases} 2^{M_1} \cdot \{ N_1 + (\overline{N_2} + 1) \cdot 2^{-(M_1 - M_2)} \} & M_1 > M_2 \\ 2^{M_2} \cdot \{ N_1 \cdot 2^{-(M_2 - M_1)} + (\overline{N_2} + 1) \} & M_1 \leqq M_3 \end{cases}
\end{aligned}
$$

The operation of the above parenthesized terms using the HSP's ALU results in:

$$= \begin{cases} 2^{M_1} \cdot \left(N_1 + \overline{N_2} \cdot 2^{-(M_1 - M_2)} \right) & M_1 > M_2 \\ 2^{M_2} \cdot \left(N_1 \cdot 2^{-(M_2 - M_1)} + \overline{N_2} + 1 \right) & M_1 \leqq M_2 \end{cases}$$

If M1 is greater than M2, the addition of $2^{-(M1-M2)}$ is not carried out in the operation of the mantissa, causing an error.

Errors during arithmetic floating point operation, if the number of repeated products/sums reaches several thousands to several 10 thousands, may be accumulated.

If such an error is serious, the input from the multiplier to the FALU should be normalized insofar as possible (operated after storage into the ACC), and the addition should be substituted for the subtraction, after NEG (Negate) operation is carried out.

(3) Data Format Transformation

In the signal processing application system, the input data from the A/D converter and the output data to the D/A converter, in general, are of the fixed point data format. Inside the HSP, arithmetic operation is conducted in the floating point data format to implement high accuracy. (See Fig. 3-5.)

To efficiently convert the data format, the HSP is capable of interchanging the floating point data format and the fixed point data format with each other through the execution of a one-step instruction. The instruction provides transformation using a transformation scaling factor on a data memory (ROM or RAM).

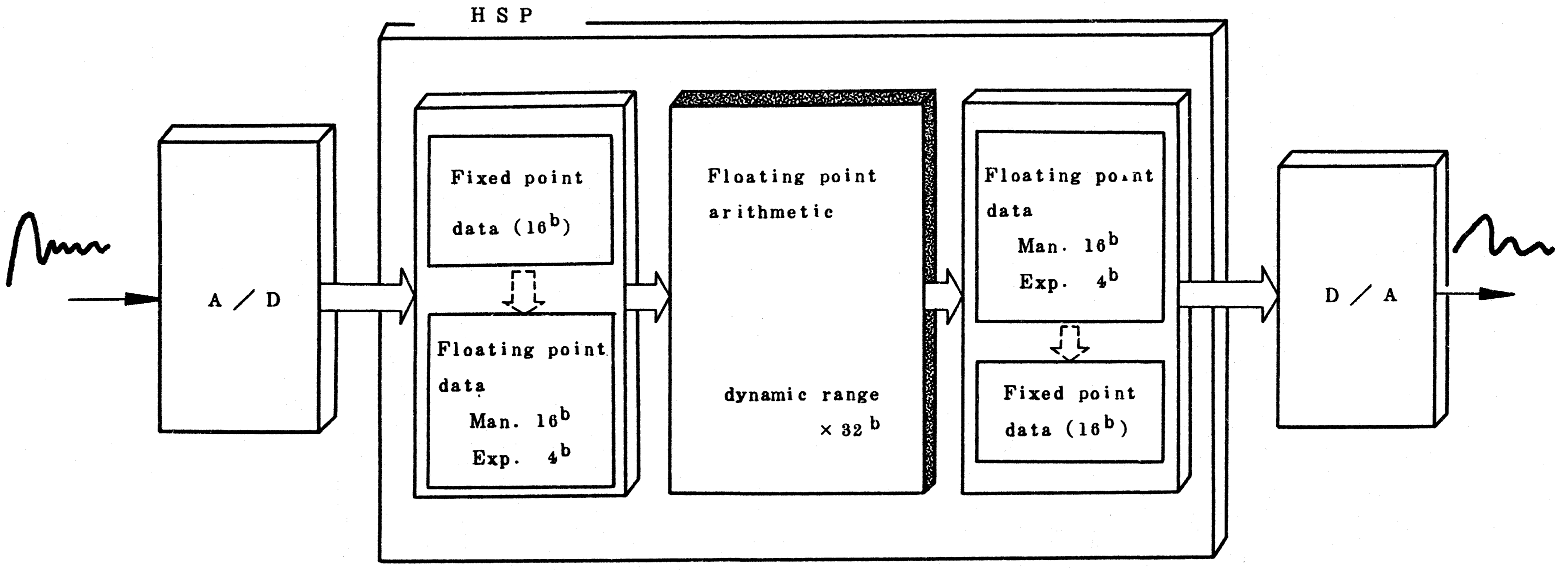

HSP
A / D
Fixed point data (16^b)
Floating point data Man. 16^b Exp. 4^b
Floating point arithmetic
dynamic range × 32 b
Floating po.nt data Man. 16^b Exp. 4^b
Fixed point data (16^b)
D / A

Fig. 3-5 Data Format Transformation

(i) From fixed point data to floating point data (See Fig. 3-6.)

$A \times 2^{n} = A1 \times 2^{n1}$ (only for normalizing operation)

A: Entered fixed point data

n: Scaling factor (in data memory)

A1: Mantissa after transformation

n1: Exponent after transformation

The floating point transforming data is basically $A \times 2^{n}$, and $A1 \times 2^{n1}$ obtained by normalizing A is finally stored into an accumulator.

(ii) From floating point to fixed point (See Fig. 3-7.)

$B \times 2^{m} + 0 \times 2^{\ell} = B1 \times 2^{\ell}$

B: Mantissa before transformation

m: Exponent before transformation

ℓ: Scaling factor (in data memory)
Factor after transformation (unnecessary)

B1: Fixed point data after transformation

Data of $B \times 2^{m}$ is normalized at 2^{ℓ} for output. If ℓ is equal to or greater than m, the mantissa is arithmetically right-shifted by ℓ-m, resulting in B1. On the contrary, if ℓ is less than m, the mantissa of B is arithmetically left-shifted by m-ℓ, resulting in B1. If an overflow occurs, however, the maximum positive or negative value is automatically assigned to B1.

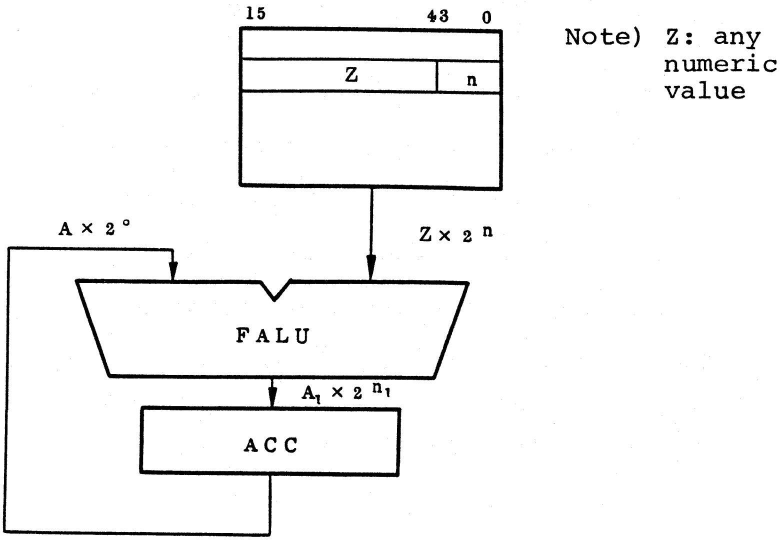


Fig. 3-6 Data Format Transformation Operation (From Fixed Point to Floating Point)

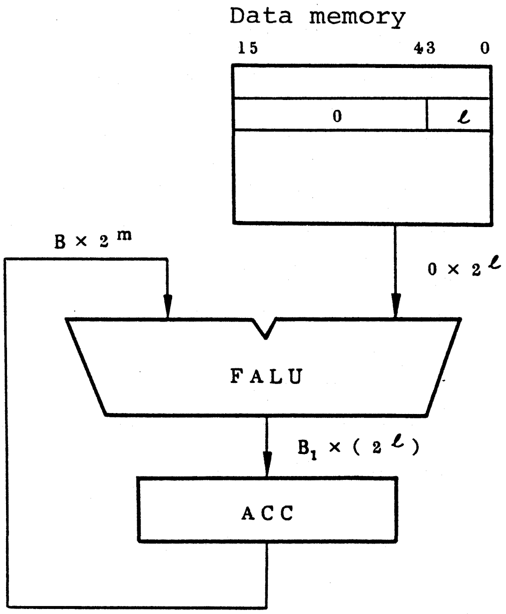


Fig. 3-7 Data Format Transformation Operation (From Floating Point to Fixed Point)

3.4 Floating Point Multiplication

(1) Multiplier Configuration

The HSP's multiplier is composed of a section which multiplies the mantissa and a section which adds the exponent part for execution of floating point operation. The I/O data configuration is:

Mantissa: 12 bits x 12 bits → 16 bits

Exponent part: 4 bits + 4 bits → 4 bits

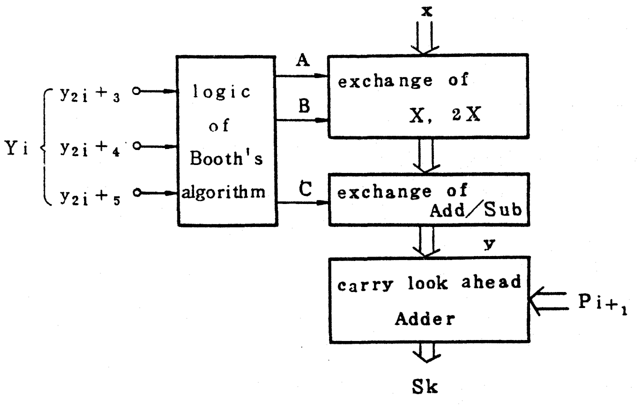
The secondary Booth algorithm is used to multiply the mantissa part.

The following is the fundamental expression:

$$
\begin{aligned}
Z &= X \cdot Y \\
&= \sum_{i=0}^{5} (y_{2i+3} + y_{2i+4} - 2 y_{2i+5}) \cdot X \cdot 2^{2i} \\
&= \sum_{i=0}^{5} P_i \cdot 2^{2i}
\end{aligned}
$$

$$S_R = X \cdot y_{i+4} + P_{i+1}$$

Fig. 3-8 shows the basic multiplier configuration. Fig. 3-9 shows the whole floating point multiplier block diagram.



$$A = y_{2i+4} \oplus y_{2i+3}$$

$$B = \overline{y_{2i+5}} \cdot y_{2i+4} \cdot y_{2i+3} + y_{2i+5} \cdot \overline{y_{2i+4}} \cdot \overline{y_{2i+3}}$$

$$C = y_{2i+5}$$

Fig. 3-8 Basic Multiplier Block Diagram

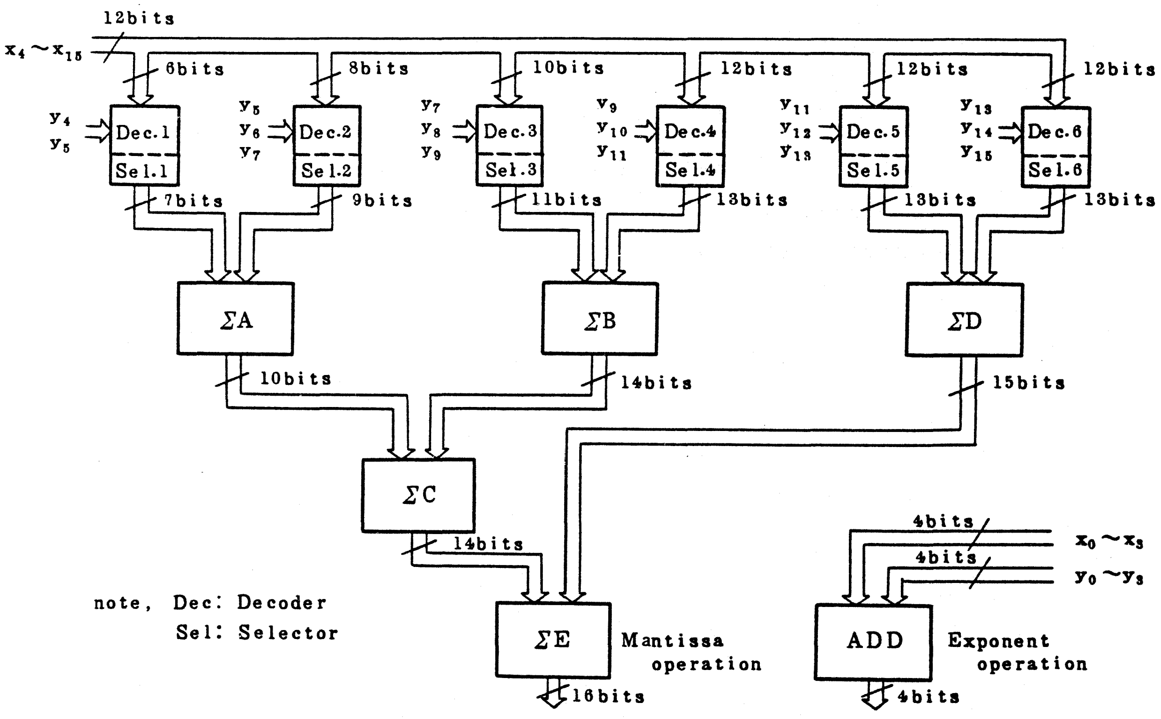


Fig. 3-9 Floating Point Multiplier Block Diagram

(2) Overflow/Underflow Protect Function

The HSP's multiplier protects three items of an exponent part's overflow and underflow and mantissa'a overflow. For floating point multiplication, the mantissa must have been normalized, if the exponent parts is greater than -8. Unless this condition is met, this will cause an error.

(i) Exponent part's overflow protection

Assume that the addition of the exponent part during the multiplication of floating point results in the exponent equal to or greater than +8.

(a) Unless the mantissa is normalized, the mantissa is 1-bit shifted to the left direction and the exponent part is made to be +7, if the exponent part is equal to +8.

If the exponent part is greater than +8, the mantissa is made to be the maximum absolute value (the sign is not changed.) and the exponent part is made to be equal to +7.

(b) If the mantissa is normalized, the exponent part is made to be equal to +7, and the mantissa is made to be the maximum absolute value (the sign is not changed.).

(ii) Exponent part's underflow protection

If floating point multiplication provides an exponent's value n of -8 or less, protection is accomplished as follows:

The exponent is fixed at -8, and the mantissa is shifted to the lower value direction by the number of bits corresponding to a value of (-8-n). For the items (ii), see processing in the ALU section.

(iii) Mantissa's overflow

In the multiplication of the HSP's mantissa, the weight of the most significant bit may be made to be -2^0, if the significant bit length of an output value is required to be greater. If two input data are -1, the result is +1, which cannot be expressed by the HSP.

For (-1) x (-1), a correction is made so that the mantissa may be approximated using the maximum positive value $(1 - 2^{-15})$.

(3) Multilier's Error

The HSP does not operate a partial product placed at the lower six bits during the multiplication of the mantissa. The configuration is such that the upper 16 bits are output as the result. (See Fig. 3-10.) As a result, an error of $2^{-1/16}$ or $2^{-2/16}$ may be caused vis-à-vis the operation result obtained by rounding the leat significant bit between 0 and 1 at 17 bit output configuration multiplication.

Even though the X input is $000, the multiplication result may not be $000, unless the Y input is $000. If the Y input is $000, the multiplication result is $000 for any value of the X input.

(4) Other Notes

When entering the multiplication result into the adder/ subtractor in the next instruction cycle, the previous cycle must be multiplied in the floating point operation mode, if the floating point mode is applied to that addition/subtraction. If the fixed point mode is applied to addition/subtraction, multiplication must be also conducted in the fixed point mode.

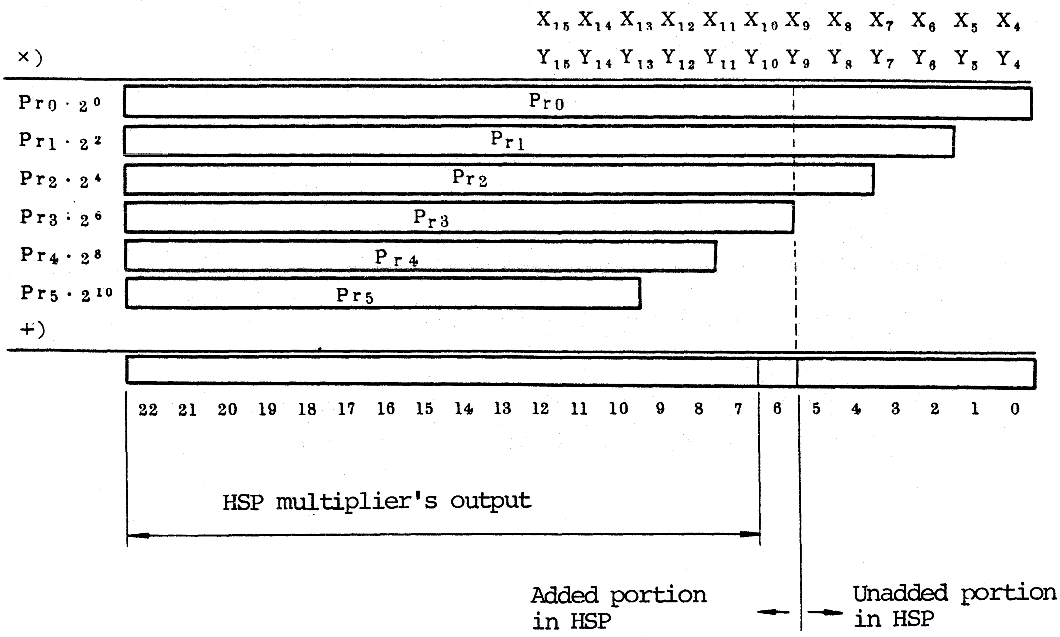


Fig. 3-10 Multiplier's Error

4. DATA MEMORY

4.1 Configuration

To improve the throughput of product sum operations, the HSP is provided with a data RAM of 200 words x 16 bits (for data), a data ROM of 128 words x 16 bits (for coefficients), and four general registers (GRs) that serve as working registers. From these memories, data of 2 words is transferred at the same time to the multiplier (FMULT) and the adder (FALU) through two buses (X and Y). Data is written into the RAM and GRs through the data bus.

The memory configuration is shown in Fig. 4-1.

The data RAM and the data ROM are partitioned into pages. Page partitioning and address assignment are detailed in Fig. 4-2.

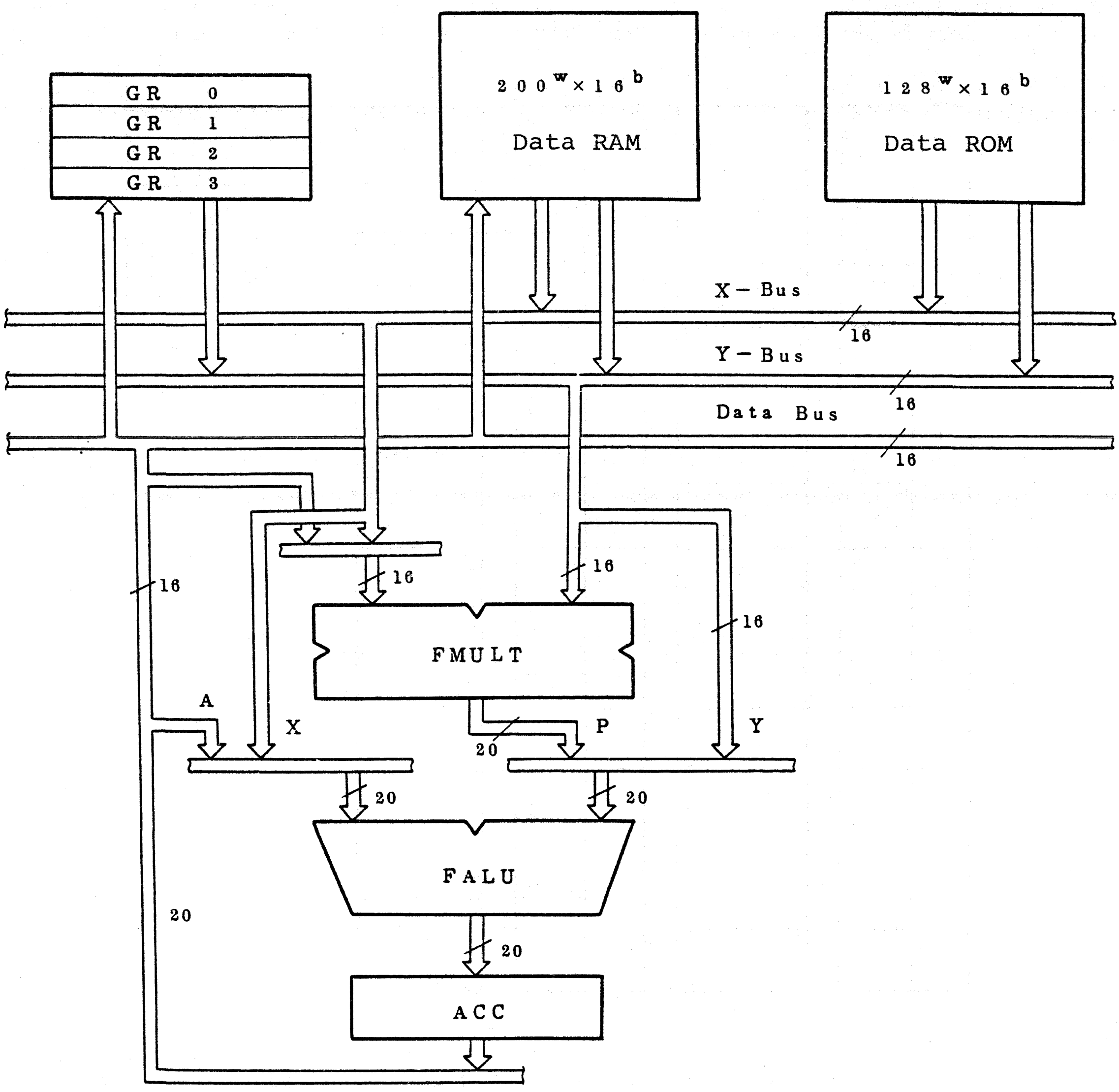
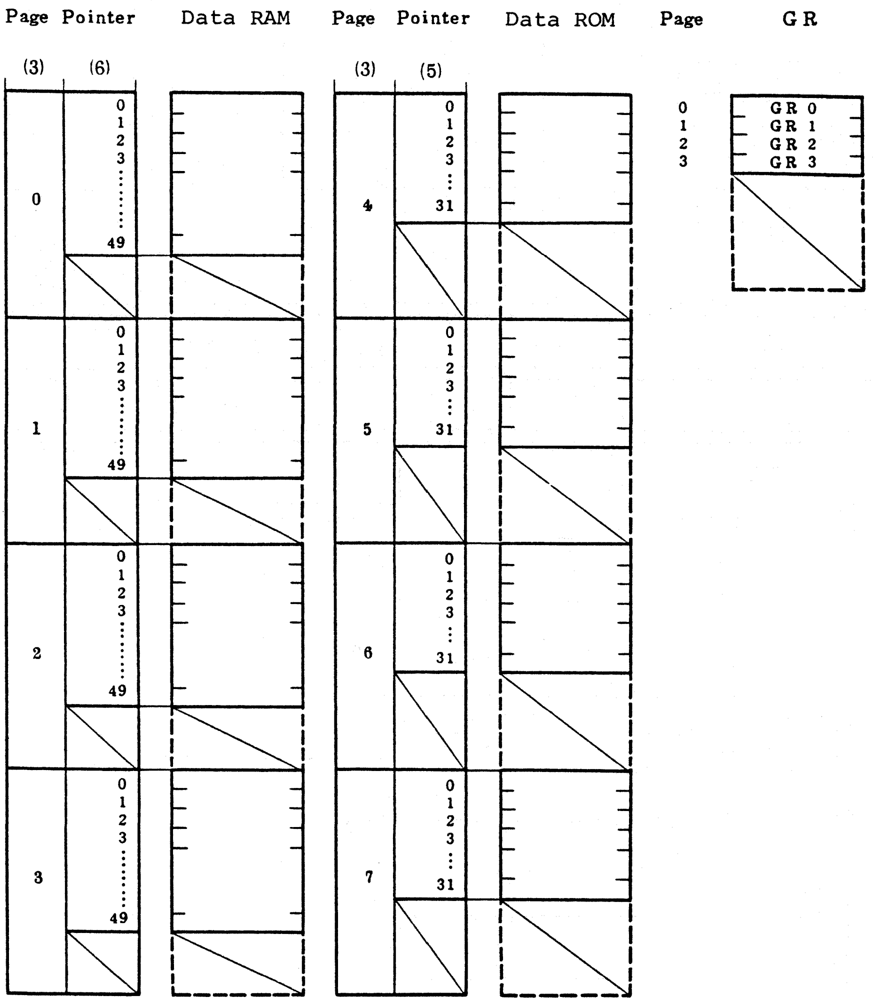


Fig. 4-1 Memory Configuration



Note: The figure in ( ) means the count of address bits.

Fig. 4-2 Data Memory Configuration and Address Assignment

4.2 Data RAM

The data RAM has a size of 200 words x 16 bits. As shown in Fig. 4-2, the RAM is partitioned into four pages (page addresses are 0 to 3), each of which consists of 50 words. The RAM is of 2-port accessible structure. This permits different data items to be read at the same time from different pages if the pointer addressing mode explained later is employed.

In the pointer addressing mode, data is read out through a combination of the 6-bit address of the RAM pointer and the two 3-bit addresses of the instruction code. In this case, the address in each page (or the pointer address) has a common value because they use the output of a single RAM pointer. This is detailed in Fig. 4-3. Output goes to the X bus from the page address selected in the X-page part of an instruction; and to the Y bus from the page address selected in the Y-page part.

Through the data bus, data is written into the RAM; that is, into the address determined by a combination of the RAM pointer and the Y-page. Write data to the data RAM comes from the ACC and the DREG.

In the direct addressing mode, data of 9 bits contained in the address part of the instruction code comes to the Y-page and RAM pointer input line (shown in Fig. 4-3), where the address of a single page is selected and one word is found out.

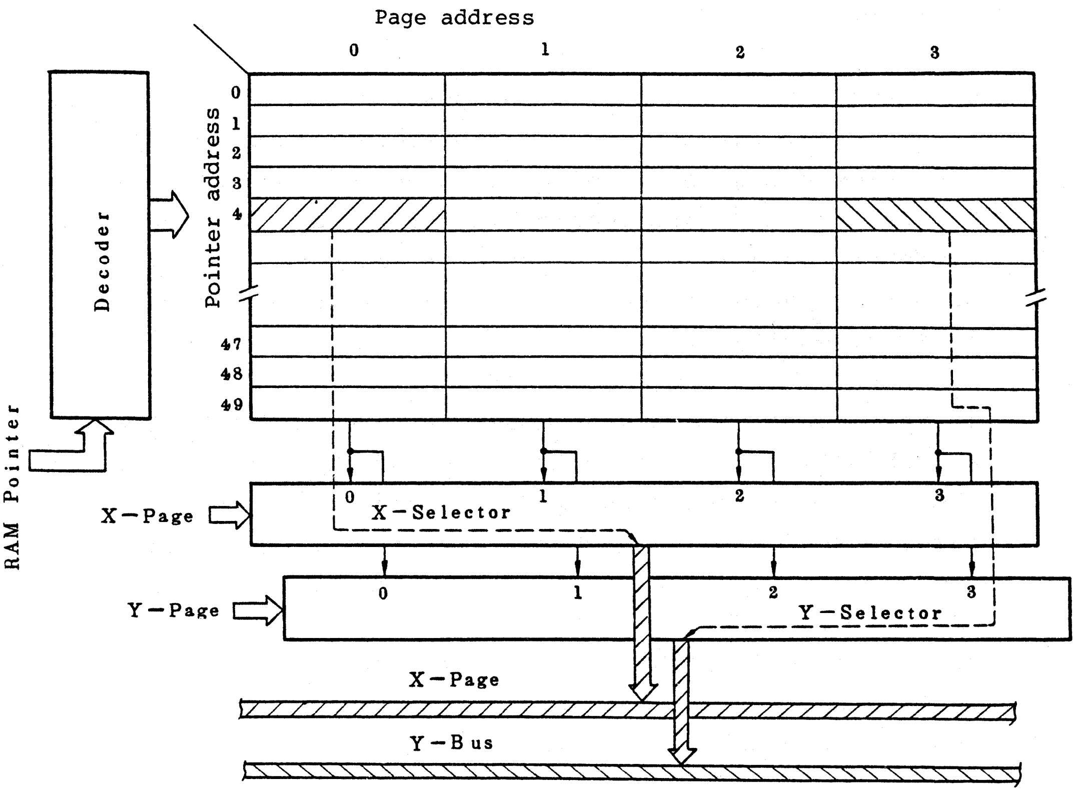


Fig. 4-3 Data RAM Access Method
(Pointer Addressing Mode)

4.3 Data ROM

The data ROM, as shown in Fig. 4-2, has a size of 128 words x 16 bits. As with the RAM, the ROM is partitioned into four pages (page addresses are 4 to 7). The page addressing mode generates effective addresses through combining an instruction's X/Y page address part (each 3 bits) and the ROM pointer (6 bits). The data ROM is different from the RAM in that only one word of data may be read out at a time. However, it is possible to send the same data to the X bus and Y bus, as shown in Fig. 4-4. It should be noted that the same page address must be used if the page address of the X-page and that of the Y-page indicate page addresses of the data ROM.

4.4 Memory Addressing Mode

The HSP provides two major memory addressing modes. They are pointer addressing mode and direct addressing mode.

The pointer addressing mode generates effective addresses by combining the value in the RAM/ROM pointer and the page address part in the instruction code. This mode may be effectively used for gaining access to data of two words during product sum operations or for reading repeated data from successive addresses. This pointer addressing mode includes a mode which provide access to ROM/RAM data and GR data at the same time.

The direct addressing mode handles the value of the 9-bit address part of the instruction code as an effective address.

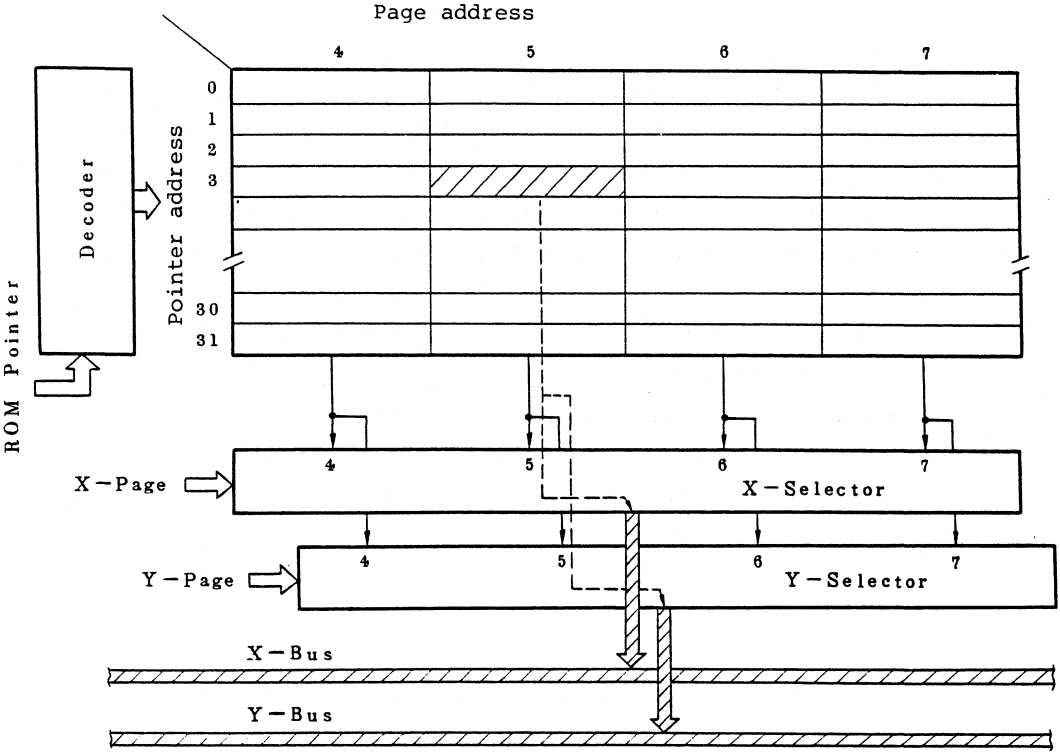


Fig. 4-4 Data ROM Access Method (Pointer Addressing Mode)

The direct addressing mode is suitable for multiplication of one-word data in the ROM/RAM memory and data in the ACC or for reading out data of discrete addresses in the memory.

The above discussion is summarized in Table 4-1.

Table 4-1 Addressing Modes

Addressing Mode	Symbol	Effective Addressing		Multiplier Operand	Instruction	Note
		X	Y/G			
Pointer Addressing	X·Y	<RAM>=(RAM Pointer A/B) +(X-Page) <ROM>=(ROM Pointer A/B) +(X-Page)	<RAM>=(RAM Pointer A/B) +(Y-Page) <ROM>=(ROM Pointer) +(Y-Page)	P=X·Y	ALU operation NOP Repeat etc.	Combination of the bus output memory RAM(X)–RAM(Y) RAM(X)–ROM(Y) } Possible RAM(X)–ROM(Y) ROM(X)–ROM(Y)XPage=Y Page
	X·G	<RAM>=(RAM Pointer A/B +(X-Page) <ROM>=(ROM Pointer) +(X-Page)	(GR)=(Y-Page)	P=X·G		RAM(X)–GR } Possible ROM(X)–GR
Direct Addressing	D	—	<RAM>/<ROM> =Inst(Direct Address)	P=ACC·Y	ALU operation NOP Repeat etc.	

(Note) <RAM> <ROM>) Address

(Note) P = Product

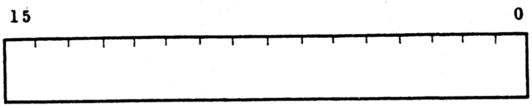
4.5 Memory Data Format

Figure 4-5 shows the formats of data stored in the data ROM, data RAM and GRs.

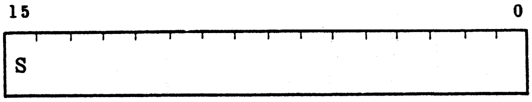
It should be remembered, for floating point data, that only the upper 12 bits of the ACC's 16-bit mantissa part are stored.

(1) Fixed point data (16 bits)

° Binary



° 2's complement



(2) Floating point data (16 bits)

° 2's complement

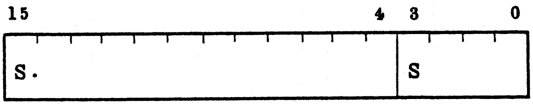


Fig. 4-5 Data Format in Data Memory

5. INTERNAL REGISTERS

5.1 Accumulator (ACC)

The HSP has two 20-bit accumulators (ACCA and ACCB). Either ACCA or ACCB may be freely selected by setting the accumulator select bit in the instruction code.

Figures 5-1 and 5-2 each show the input/output data of the accumulator available when the mode of operation is fixed and floating point representations.

When it is necessary to store floating point data which consists of 16 bits for its mantissa and 4 bits for its exponent, use either ACCA or ACCB as a storage accumulator; or store the data in the 2-word portion of the data memory (RAM or GR).

This may be done by the following approach:

(1) To store accumulator data

(a) Store accumulator data of fixed point representation in RAM1.

(b) Using floating point representation, store the same data in RAM2.

(2) To transfer the contents of data RAM to the accumulator

(a) Store the contents of data RAM1 in the accumulator.

(b) By regarding the contents of data RAM2 as a scaling constant, convert the accumulator data from fixed to floating point representation.

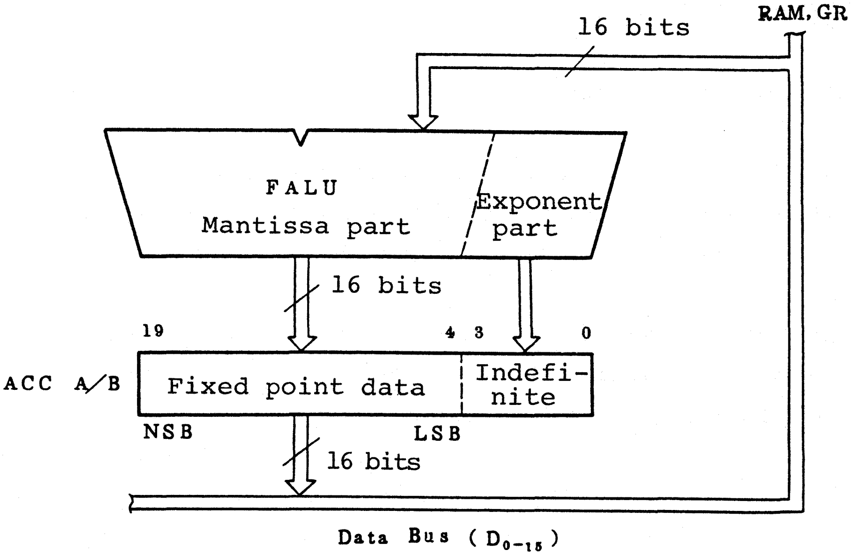


Fig. 5-1 Accumulator Input/Output Data
(Fixed Point Arithmetic Operation)

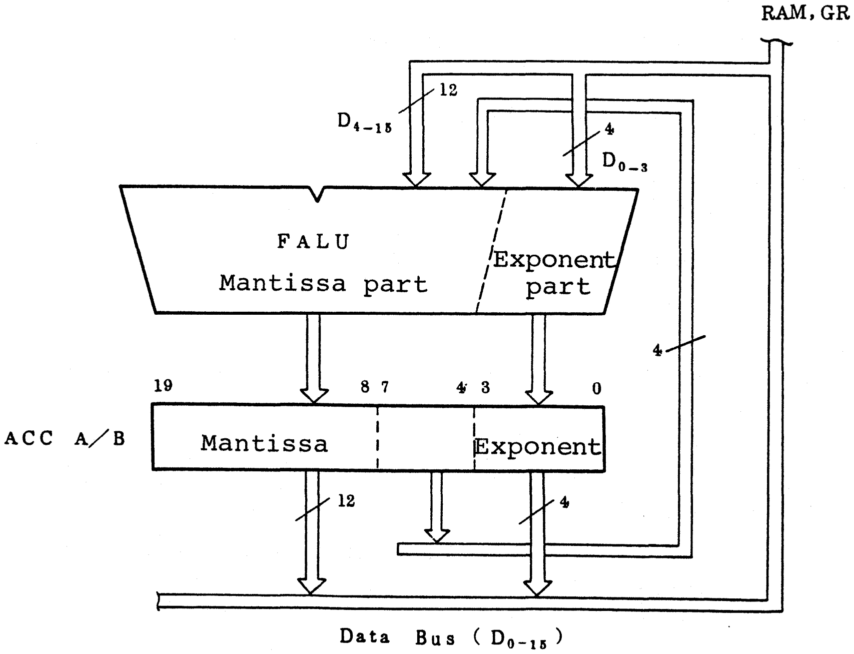
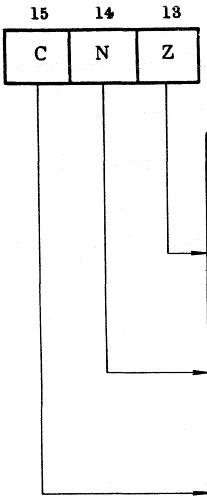


Fig. 5-2 Accumulator Input/Output Data
(Floating Point Arithmetic Operation)

5.2 Condition Code Register (CCR)

Each flag in the CCR reflects the result of arithmetic operation in the ALU of the FALU. There are three different flags: zero flag (Z), negative flag (N) and carry (C).

These flags correspond to the bit positions D13 to D15 of the data bus. The contents may be transferred between the CCR and the ACC in response to an instruction.



Z, N, C : Affected by fixed point arithmetic operation

Z, N : Affected by floating point arithmetic operation

Flag	Set/Clear condition
Zero flag	Cleared (0) when $M \neq 0$ Set (1) when $M = 0$
Negative flag	Cleared (0) when $M \geqq 0$ Set (1) when $M < 0$
Carry flag	Cleared (0) when $C = 0$ Set (1) when $C \neq 0$

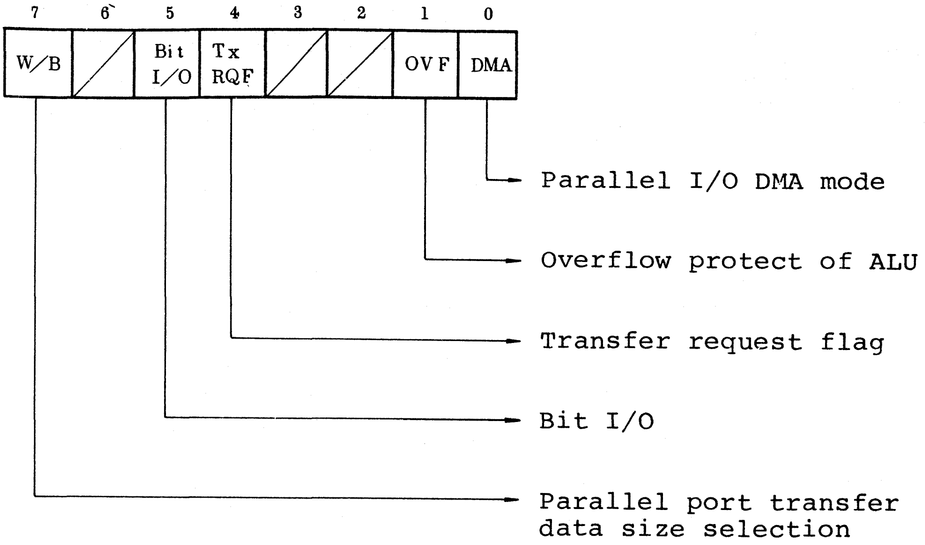
Note: M = Mantissa part after arithmetic operation
C = ALU's carry after arithmetic operation

Fig. 5-3 CCR Set/Reset Condition

5.3 Control Register (CTR)

Data from the ACC may be transferred to the control register by an instruction from the HSP. It is also possible to transfer data from the parallel ports D0 - D15. It should be noted that the condition of each flag remains indefinite during the HSP's reset operation.

The internal data bus D0 - D15 is connected to bits 0 - 7.



Note: See the next page for the contents of each flag.

Fig. 5-4 Control Register Functions

Bit 0 -- Parallel I/O DMA mode

1 : DMA data transfer mode
0 : Non-DMA data transfer mode

A microcomputer system outside the HSP operates in the DMA mode. Inside the HSP, however, data is transferred between the parallel I/O port and the ACC under program control.

Bit 1 -- Overflow protect of ALU

1 : Fixes the mantissa part to its greatest absolute value if there is an overflow in the mantissa of the ALU.
0 : Accomplishes no overflow protection.

Bit 4 -- Transfer request flag

In the DMA mode, this flag is set for a request of data transfer during a transfer cycle of each data element.
In response to an entry of TxAK, this flag is reset and again set automatically. Entry of a DEND signal does not cause the flag to be set.
In the non-DMA mode, this flag is effective as a programmable output.

Bit 5 -- Bit I/O

Connected to the Bit I/O terminal. Has both of the input and output functions.
For input, this flag is set to a 1 before information of the bit I/O terminal is entered directly into the data bus.

Bit 7 -- Parallel port transfer data size selection

1 : Transfers data of 16 bits (word)
0 : Transfers data of 8 bits (byte)

5.4 Status Register (STR)

The contents of the status register may be transferred to the ACC by an internal instruction. It is also possible in the internal program to transfer the value of the ACC to some flags (UF, I_{SO}, I_{SI}, I_P, I_M).

The flags SOF, SIF and PF are reset when the contents of the STR are transferred to the ACC by an internal instruction.

The status register is connected to the internal data bus D0 - D7.

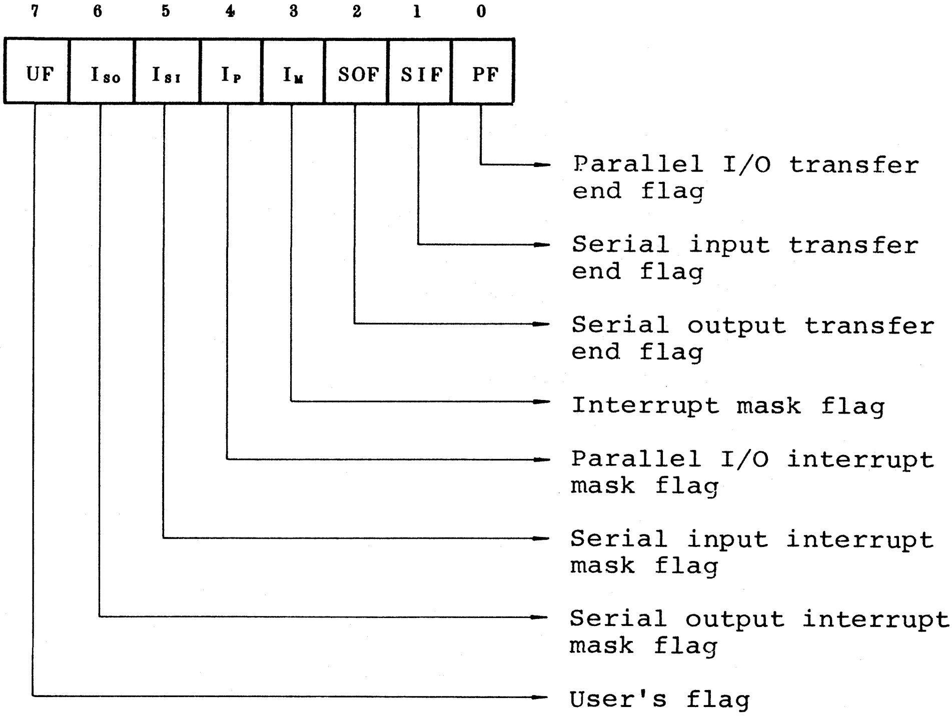

7
6
5
4
3
2
1
0
UF
I_{SO}
I_{SI}
I_P
I_M
SOF
SIF
PF
Parallel I/O transfer end flag
Serial input transfer end flag
Serial output transfer end flag
Interrupt mask flag
Parallel I/O interrupt mask flag
Serial input interrupt mask flag
Serial output interrupt mask flag
User's flag

Fig. 5-5 Status Register Functions

Bit 0 -- Parallel I/O transfer end flag

This flag is set at the trailing edge of IE, or after the end of data transfer in D0 - D15. (In the byte transfer mode, the flag is set at the trailing edge after the higher byte has been transferred.)
The flag is set also when an interrupt disable condition is established.
Under an interrupt enable condition, if the flag is set, an interrupt occurs in the HSP.
The flag is reset when data is transferred from STR to ACC.

Bit 1 -- Serial input transfer end flag

This flag is set after the end of serial input data transfer.
The flag is set also when an interrupt disable condition is established.
Under an interrupt enable condition, if the flag is set, an interrupt occurs in the HSP.
The flag is reset when data is transferred from STR to ACC.

Bit 2 -- Serial output transfer end flag

This flag is set after the end of serial output data transfer.
The flag is set also when an interrupt disable condition is established.
Under an interrupt enable condition, if the flag is set, an interrupt occurs in the HSP.
The flag is reset when data is transferred from STR to ACC.

Bit 3 -- Interrupt mask flag

This flag masks the interrupt of every factor.
The flag may be set/reset also by an instruction.
It is automatically set when an interrupt has taken place; and reset by an RTI instruction.
The flag is a 1 under a mask condition; and a 0 under a non-mask condition.

Bit 4 -- Parallel I/O interrupt mask flag

This flag masks an interrupt generated by a parallel I/O (when the PF is set).
The flag is set/reset by an instruction.
It is a 1 under a mask condition; and a 0 under a non-mask condition.

Bit 5 -- Serial input interrupt mask flag

This flag masks an interrupt generated by serial input (when the SIF is set).
The flag is set/reset by an instruction.
It is a 1 under a mask condition; and a 0 under a non-mask condition.

Bit 6 -- Serial output interrupt mask flag

This flag masks an interrupt generated by serial output (when the SOF is set).
The flag is set/reset by an instruction.
It is a 1 under a mask condition; and a 0 under a non-mask condition.

Bit 7 -- User's flag

This flag may be used at will by the user.
The flag is set/reset by an instruction.

5.5 Repeat Counter (RC)

The repeat counter is used mainly for repeated program operations.

The individual bits of this counter are connected to bits 10 - 15 of the data bus; and this should be remembered when data is transferred from the ACC.

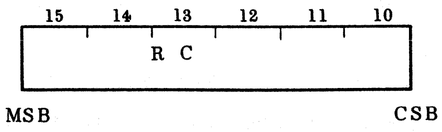


Fig. 5-6 Repeat Counter

Here are examples in which repeated operations are used in this counter.

(1) Repeated operations by repeat instruction

Step 1 RC --> #n
where the "n" indicates the number of repetitions.

Step 2 Repeat instruction
which allows the next instruction to be repeated.

Step 3 Arithmetic instruction (RC - 1)
which enables n+1 operations.

(2) Repeated operations by jump instruction

Step 1 RC --> #n
Step 2 Instruction 1
Step 3 Instruction 2
Step 4 Jump to step 2 if (RC) ≠ 0 and (RC) - 1
Step 5 ......

Example (2) repeats instructions 1 and 2 until the value of the RC reaches a zero, and advances to step 5 when (RC) = 0 is reached.

Note that this repeat counter is automatically decremented when an ACC arithmetic operation has incremented the RAM pointer or ROM pointer.

5.6 Address Pointer (RAM Pointer A/B, ROM Pointer)

The data memory has three address pointers: two for RAM and one for ROM. Each of the pointers is connected to bits 10 - 15; and this should be remembered when address data is transferred from the ACC. Figure 5-8 shows how to generate effective addresses for the data RAM and data ROM by means of the address pointers.

A combination of the instruction code's page address part (X-page and Y-page) and these address pointers generates 9-bit effective addresses.

The RAM reads out 2-word data at the same time with the help of two page addresses and one pointer address. Data is written into addresses generated by the Y-page and RAM pointer.

The ROM can read out only one word at a time. Therefore, any X-page and Y-page addresses in the ROM area must be contained in the same page.

An instruction enables selection between the RAM pointer A and the RAM pointer B. It should be noted that each pointer may be automatically incremented during the same instruction cycle as memory access.

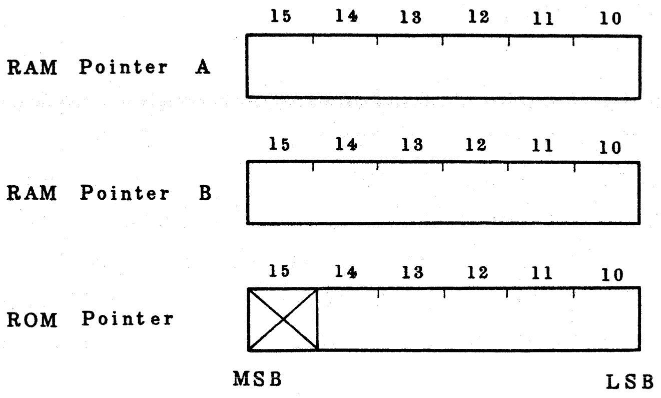
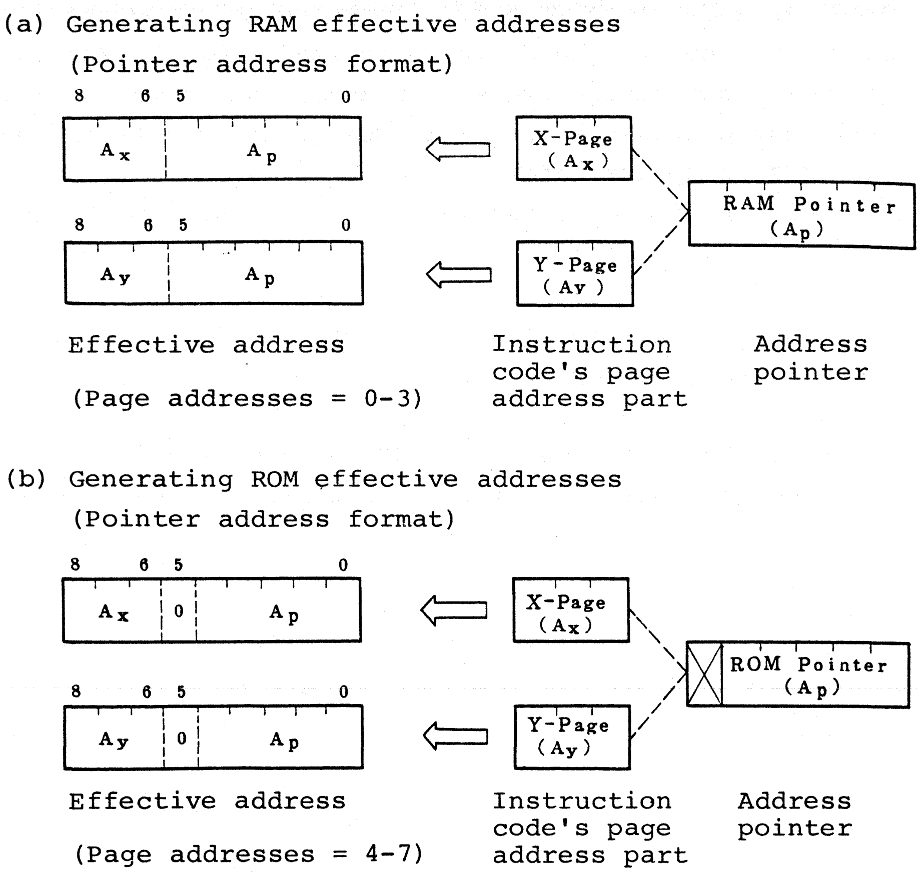


Fig. 5-7 Address Pointers


(a) Generating RAM effective addresses (Pointer address format)

(b) Generating ROM effective addresses (Pointer address format)

Fig. 5-8 Generation of Effective Addresses

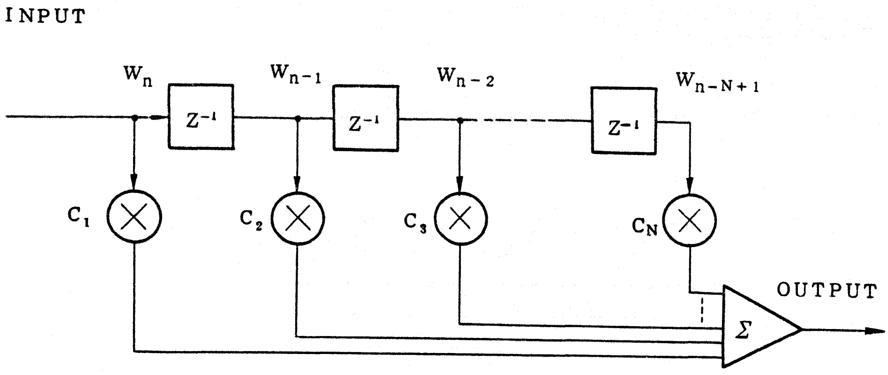
5.7 Delay Register (DREG)

The arithmetic operation for signal processing requires, for example, a transversal filter (as shown in Fig. 5-9) which has a function (equivalent to Z^{-1}) of delaying an input data string by one sample for each sampling cycle. For the HSP in which input data is arranged on RAM, it is necessary to effectively implement the function that delays the data storage by one address for each sampling cycle.

For this reason, the output of the RAM is provided with a DREG, as shown in Fig. 5-10. Composed of a 2-stage latch ciurcuit, the DREG holds the data read from the RAM until the next instruction cycle is reached, as shown in Fig. 5-11; and writes the data into memory after data is read out from an updated address. When repeated, this operation will shift addresses in the data group memory, as shown in Fig. 5-11(b).

Writing from DREG to RAM may be controlled by the description in the instruction code.

It should be noted that the DREG permits data to be exchanged not only with the RAM but with the GR as well.



$$Q_n = \sum_{i=1}^{N} Ci \times W_{n-i+1} \qquad W_i \rightarrow W_{i-1}$$

Fig. 5-9 Transversal Filter

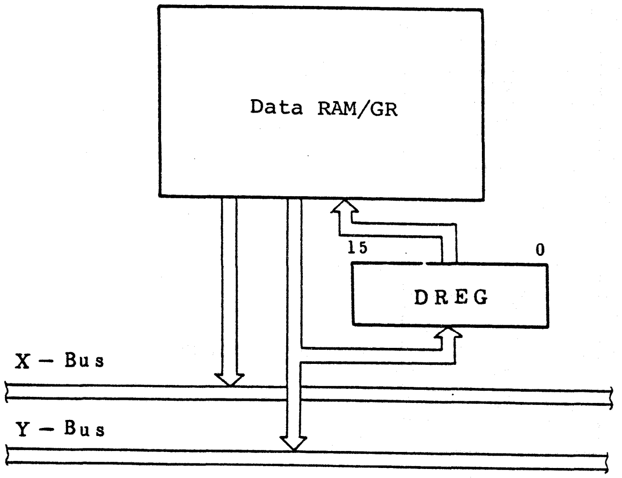
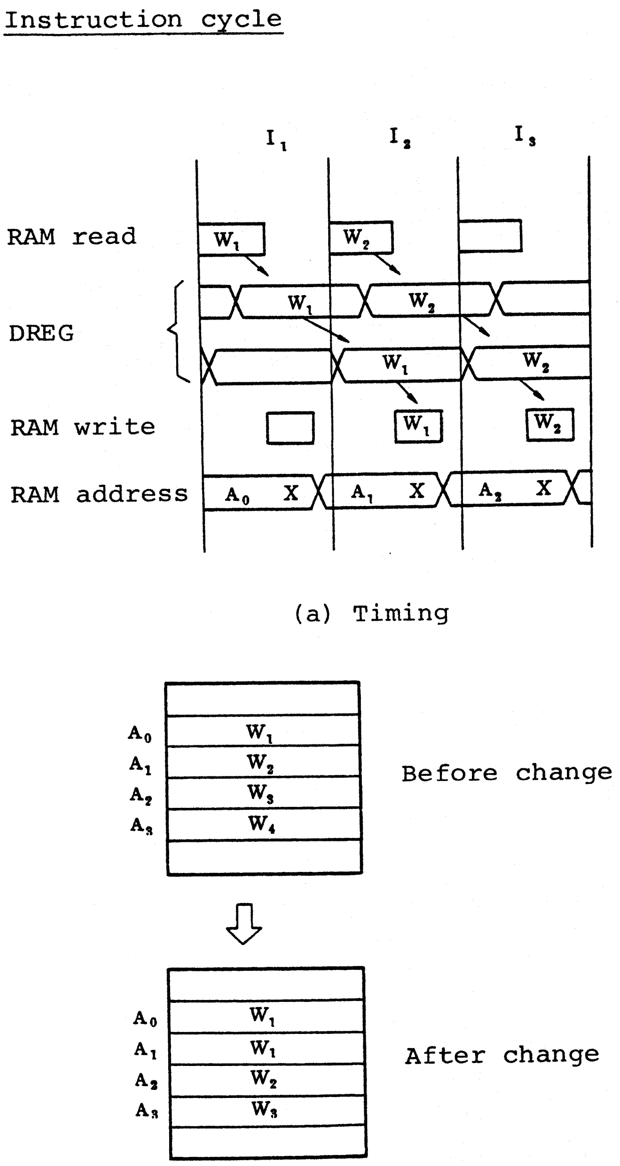


Fig. 5-10 Location of DREG



(a) Timing

A0	W1
A1	W2
A2	W3
A3	W4

Before change

A0	W1
A1	W1
A2	W2
A3	W3

After change

(b) Changing Memory Contents

Fig. 5-11 DREG-Based Shifting of RAM Data Addresses

6. INPUT/OUTPUT INTERFACE

6.1 Function Control

The HSP uses input information (code) at input terminals F0-F3 to transfer data between the inside of the HSP and the outside bus through parallel ports D0-D15.

Input information at F0-F3 is effective when input terminal $\overline{CS}$ is active. A part of this function control works to halt the clock working in the HSP. As shown in Fig. 6.1, the HSP gets into a stop mode when an instruction cycle is over after function information is internally detected. The stop mode is released after a change in function information is detected, and this is followed by the execution of the subsequent instructions to be left behind before the stop action began.

Because of it dynamical operation, the HSP must internally deal with halts of 10 microseconds or less. Function information is detected in accordance with the timing of the internal clock, so that it is necessary for the function input to remain active during one or more instruction cycles.

The direction of input/output transfer is controlled by input signals at the R/W terminal.

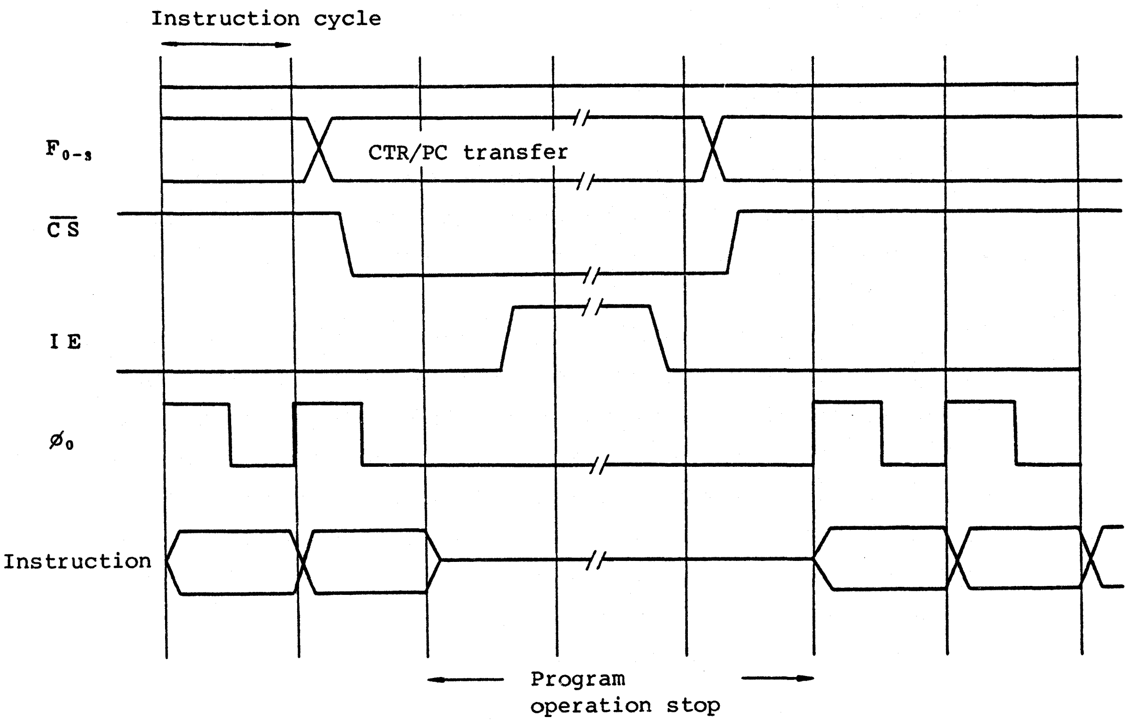


Fig. 6-1 Program Operation Stop

Table 6-1 Details of Function Control

CS	F3	F2	F1	F0	Operation [Operation mode & Interrupt]
1	*	*	*	*	No I/O operation [Operation mode] Program mode [Interrupt] --
0	0	0	0	0	No I/O operation [Operation mode] Program mode [Interrupt] --
	0	0	1	0	Data transfer (Lower byte) CTR(W/B)=0 Read : D0-D7 ←-- OR0-OR7 Write : D0-D7 --→ IR0-IR7 [Operation mode] Program mode [Interrupt] None
	0	0	1	1	Data transfer (1) Byte transfer mode CTR(W/B)=0 Upper byte transfer Read : D0-D7 ←-- OR8-OR15 Write : D0-D7 --→ IR8-IR15 (2) Word transfer mode CRT(W/B)=1 Word (16-bit) parallel transfer Read : D0-D15 ←-- OR0-OR15 Write : D0-D15 --→ IR0-IR15

-cont'd-

CS	F3 F2 F1 F0	Operation [Operation mode & Interrupt]
0	0 0 1 1	[Operation mode] Program mode [Interrupt] possible
	0 1 0 0	CTR transfer "Write" only Write : D0-D7 --> CTR0-CTR7 [Operation mode] Stop mode [Interrupt] None
	1 0 0 0	PC transfer "Write" only Write : D0-D7 --> PC0-PC7 (Setting as D10=1, D9=0 and D8=0) Set CTR's interrupt mask flag. No other register conditions are guaranteed. The contents of RAM are reserved. [Operation mode] Stop mode [Interrupt] None

6.2 Parallel Port (Microcomputer) Interface

The HSP's parallel input/output terminals D0-D15 form a bidirectional three-state bus. There are three different types of data transfer between input/output terminals D0-D15 and the inside of the HSP.

(1) Word data transfer

Input/output data at input/output terminals D0-D15, as 16-bit data, is transferred in parallel between input registers IR0-IR15 and output registers OR0-OR15.

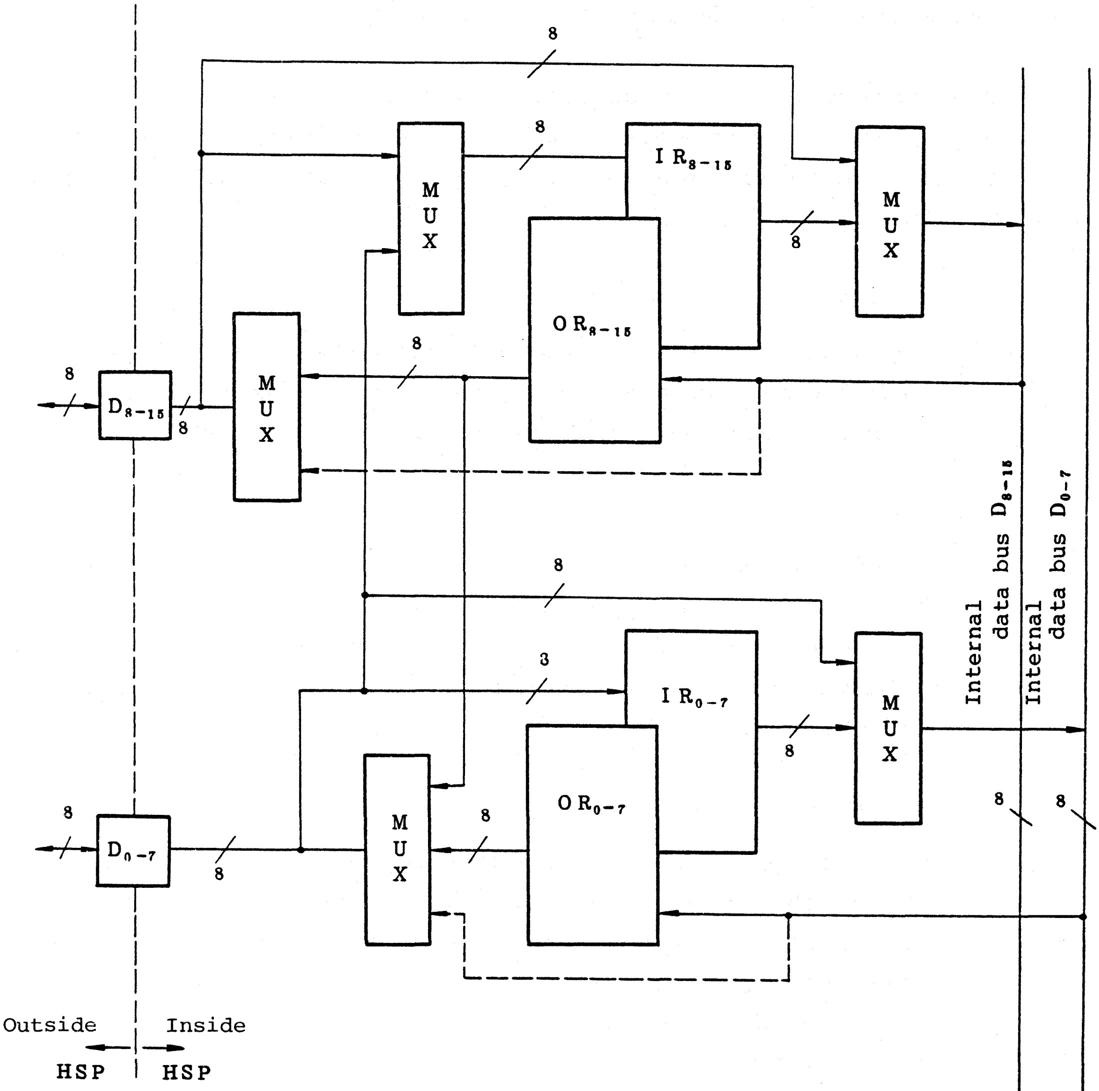
(2) Byte data transfer

Input/output data at input/output terminals D0- D7 is divided into the upper byte (IR8-IR15 & OR8-OR15) and the lower byte (IR0-IR7 & OR0-OR7) by function control information, and transferred between the IR registers and the OR registers.

(3) CTR & PC data transfer

Input/output terminals D0-D15 are connected directly to the 16-bit data bus (D0-D15) in the HSP, and data is transferred with each register in the HSP. In this case, the operation in the HSP is brought to a halt so that the HSPs data bus is connected directly to the external bus.

Data is set in each of these registers according to the IE's timing.



Note: Each dotted line denotes a signal path for testing.

Fig. 6-2 Parallel Port Interface

6.3 Serial Input/Output

The serial input/output function is intended mainly as an interface for A/D and D/A converters. The HSP provides serial input/output of up to 16 bits.

The functions of serial input and serial output are given in Table 6-2.

(1) Serial Input

A serial input takes place on an MSB first basis. Even for less than 16 bits, the internal counters are used and data is automatically shifted so that the MSB of data goes to the most significant bit of the shift register. It should be noted, however, that an SI input signal is, as it is, applied to the lower bits after the specified bit. In this case, it is necessary to enter more than 16 clocks of SICK even when the SIEN is in the non-active mode.

No data transfer to the ACC is allowed while serial input is in process. The SIR is cleared when data is transferred to the ACC.

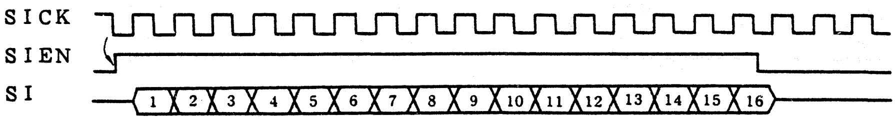
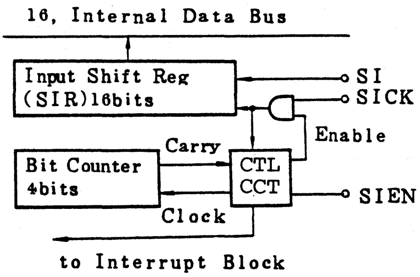
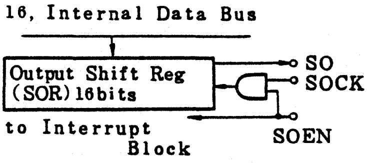
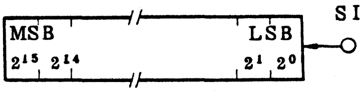
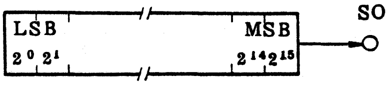


Fig. 6-3 Serial Input Timing

(2) Serial Output

The basic timing is the same as with serial input. For the HSP's serial output, "0" goes to the SOR after data is transferred to the outside. See Table 6-2 for other information.

Table 6-2 Serial Input/Output

Serial input	Serial output
[Block diagram]  16, Internal Data Bus Input Shift Reg (SIR)16bits SI SICK Enable Carry Bit Counter 4bits CTL CCT SIEN Clock to Interrupt Block	 16, Internal Data Bus Output Shift Reg (SOR) 16bits SO SOCK to Interrupt Block SOEN Terminal SO gets a three-state condition when SOEN is non-active.
[Input/output data bit numbers]  Input from MSB	 Output from MSB

-cont'd-

Serial input	Serial output
[Input/output bits control] Even if, during an active SIEN interval, there is an input of 16 or more SICK clocks, any input of data of 16 bits or more is automatically disabled. If there are less than 16 SICK clocks during an active SIEN interval, input data is shifted to the high order. However, it is necessary to enter SICK in advance.	Even if, during an active SOEN interval, there is an input of 16 or more SOCK clocks, the SO output sends out "0" for bit 16 and following bits. If there is an output of data of less than 16 bits, SOEN gets non-active when the count of bits as output has reached the specified value.
[Interrupt generation] Counting of 16 clocks starts at the leading edge of SIEN. When 16 bits of data has been entered into SIR, the internal interrupt generator circuit (SIF) is initiated.	At the trailing edge of SOEN, the internal interrupt generator circuit (SOF) is initiated.
[Shift register set/reset] After a transfer instruction (SIR→ACC) has been executed, the register is reset (all bits are 0s).	In response to a transfer instruction (ACC→SOR), data is set in the shift register.

6.4 Interrupts

The HSP is capable of generating interrupts after the end of data transfer (microcomputer interface, serial input/output) so as to effectively process internal arithmetic operations and data input/output. A schematic of the interrupt circuit is shown in Fig. 6-4.

(1) Interrupt level and factors

There is only one interrupt level. The level has three factors. They are:

(a) End of parallel port (microcomputer interface) transfer
(b) End of serial input transfer
(c) End of serial output transfer

These factors are identified by program.

(2) Masking

It is possible to mask each of the I_M, I_P, I_{SI} and I_{SO} flags in the status register.

I_M : Mask flag for all interrupts.
When there is an interrupt, the flag is set automatically, resulting in a masked condition.
The flag is reset by an RTI instruction, thus resulting in a demasked condition.
The flag may be set/reset also by a transfer instruction (ACC → STR).
1 = Mask; 0 = Nonmask

I_P : Mask flag for an interrupt at the end of parallel port transfer.
The flag is set/reset by a transfer instruction.
1 = Mask; 0 = Nonmask

I_{SI} : Mask flag for an interrupt at the end of serial input transfer.
The flag is set/reset by a transfer instruction.
1 = Mask; 0 = Nonmask

I_{SO} : Mask flag for an interrupt at the end of serial output transfer.
The flag is set/reset by a transfer instruction.
1 = Mask; 0 = Nonmask

Even under these interrupt mask conditions, each of the input flags PF, SIF and SOF is set by an external input.

(3) Stack

The program counter (PC) has two stacks. Therefore, a 2-level nesting for interrupts or subroutines is possible.
There are two ACCs (ACCA and ACCB). One may be used for the main program; and the other, for an interrupt program.
Programming is made so that the other registers are saved in the RAM. It should be remembered that, if data is transferred in the floating point mode when it is stored into RAM, then the lower 4 bits of the ACC's mantissa part encounter an error.

(4) Interrupt wait

At the execution of a repeat instruction or any instruction initiated by a repeat instruction, or during the execution of a jump instruction (only for jump action) or an RTN or RTI instruction, the initiation of an interrupt has to wait until the above instruction comes to an end.

(5) Vectoring

When an interrupt occurs, the contents of address $1FF are set in the program counter. This address is the final address of the instruction ROM. Therefore, programming requires that a jump goes to the beginning of the interrupt handling program when the jump instruction is stored at the final address of the instruction ROM. An example is illustrated in Fig. 6-5.

(6) Pipeline control

Since output data of the multiplier is reserved during only one instruction cycle, if an interrupt occurs while a pipeline-based product sum operation is in process, it destroys the result of multiplication output. At a portion of a program which is carrying out an arithmetic operation under pipeline control, it is necessary to keep the part under an interrupt disable condition.

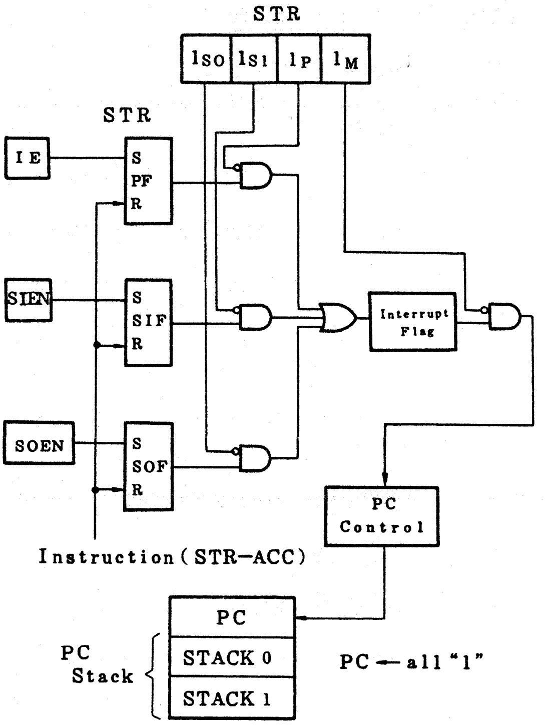


Fig. 6-4 Interrupt Circuit Configuration

```
Address    Label    Instruction

100        INT      Save ACC

                    Save CCR

                    STR --- ACC      :  Reset PF, SIF, SOF

                    ACC --- Memory   :  Save to STR memory

                    Save register    :  ROM/RAM pointer

                   ┌───────────────────────────────┐
                   │ ° Factor identification       │
                   │                               │
                   │ ° Interrupt handling program  │
                   └───────────────────────────────┘

                    Return register

                    Return CCR

                    Return ACC

                    RTI              :  Reset I_M, main
                                        routine pattern

IFF        JMP      INT
```

Fig. 6-5 Example of Interrupt Handling Program

6.5 DMA (Direct Memory Access)

6.5.1 General

The DMAC (direct memory access controller) for an 8-bit microcomputer 6800 permits data transfer between the HSP and the memory.

Through input/output terminals D0-D15, data is transferred directly, not by way of the CPU of the microcomputer, between the HSP's IR or OR and the external memory or I/O device.

Inside the HSP, the DMA flag in the CTR controls the mode as follows:

CTR(DMA) = 1 : DMA transfer mode
CTR(DMA) = 0 : Non-DMA transfer mode

This DMA transfer mode brings the microcomputer to a stop. A DMA action takes place between the IR or OR inside the HSP and the external memory or I/O device; and a program action occurs between the IR or OR inside the HSP and the ACC or memory.

The HSP's DMA mode permits the HALT burst mode as one of the DMA transfer approaches possessed by the microcomputer 6800.

Figure 6-7 shows an example of connection of control lines for the HSP, DMAC and 6800 CPU. The HSP is provided with terminals TxRQ, TxAK and $\overline{\text{DEND}}$ intended for the DMA transfer mode.

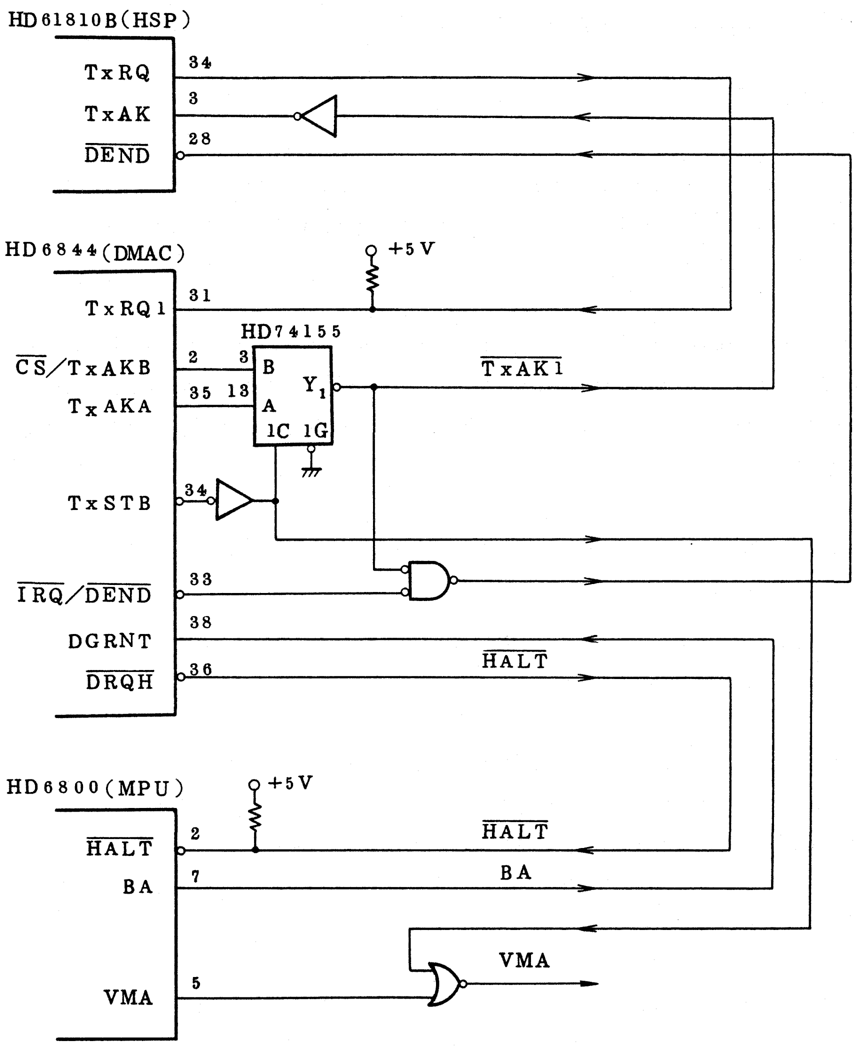


Fig. 6-6 Example of Connection between HSP and DMAC

6.5.2 Operation

Setting the CTR(DMA) to a 1 results in a DMA transfer mode. Once the CTR(DMA) has been set at a 1, it will keep the DMA mode unless it is reset to a 0 by program or DEND signal generates.

(1) Word Transfer

A DMA word transfer may be produced by setting the CTR's W/B flag to a 1 and by setting the each of the CTR's DMA flag and transfer request TxRQ flag to a 1. Reception of a TxAK signal causes the TxRQ to be a zero. In this case, the PF is set, and under an interruptable condition, the inside encounters an interrupt at the end of parallel port input/output transfer. As a result, the HSP transfers data from IR to ACC, and then to memory with the help of a program.

At the end of data transfer from IR to ACC, the TxRQ is again set to a 1 automatically, giving the DMAC a request of the next data transfer. For output from the OR, if data is read out from the OR under an interruptable condition, the PF is set and an interrupt occurs. The result is that data is transferred from ACC to OR by the HSP's interrupt handling program. When data is transferred to the OR, the TxRQ is again set to a 1, giving the DMAC a request of the next data transfer. For the final word to be transferred in the DMA mode, if it is entered together with a DEND signal, the flag for the DMA mode is reset, resulting in a non-DMA mode.

Under an interrupt disable condition, a change in the

PF flag is monitored so that the end of input/output is searched.

(2) Byte Transfer

A DMA transfer mode may be established by setting the CTR's W/B flag to a 0, the DMA flag to a 1, and the TxRQ flag to a 1. The DMA byte transfer mode provides data input/output in the order of the lower part and the upper part. Inside the HSP, after the end of a transfer of 2 bytes, data is transferred between the IR or OR and the ACC. The transfer sequence is the same as with word transfer, except that one word is covered instead of 2 bytes.

The above discussion is summarized in Table 6-3.

Table 6-3 DMA Mode Data Transfer

Mode Approach		Word transfer	Byte transfer
			8 bits
Control	DMA TxRQF	"1"	"1"
	W/B	"1"	"0"
	F0		"0": Lower / "1": Upper
Data arrangement		0 Word, 16 bits 1 Word, 16 bits 2 Word, 16 bits 3 Word, 16 bits	0 Lower 1 Upper 2 Lower 3 Upper

6.6 Bit I/O

The bit I/O may be used as shown in Fig. 6-7. The input/output signal serves as a 1-bit input/output terminal.

(1) As Input Terminal

A transfer instruction (ACC→CTR) is used to write a 1 into the corresponding bit of the CTR's bit I/O. This turns off the output MOS transistor of the bit I/O, thus allowing external data to be entered. With a transfer instruction (CTR→ACC), the input takes in data into the ACC through the data bus. Note that it is necessary to reserve input data at the terminal while the transfer instruction is being executed.

(2) As Output Terminal

Because of an open drain, the output of the bit I/O must have a pull-up resistor connected externally. This output signal may be used as, for example, as control of peripherals. It may serve also as an interrupt signal to a microcomputer.

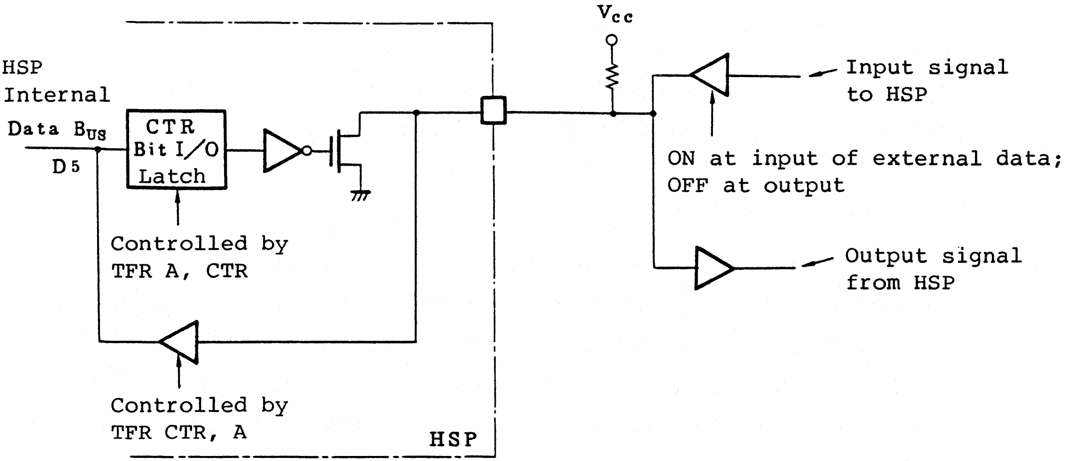


Fig. 6-7 Bit I/O Usage

7. INSTRUCTIONS

7.1 General

The instruction system of the HSP is designed so as to provide fast, efficient signal processing and arithmetic operation. The HSP has a total of 53 different instructions, covering not only arithmetic operations but also logical operations and data transfer as basic instructions for a general-purpose microcomputer.

Major features of the HSP are detailed below.

(1) Floating point arithmetic operation

The HSP, as mentioned in section 3, carries out floating point arithmetic operations, providing a 32-bit dynamic range and a 16-bit resolution (effective bit length). This provides accuracies suitable for signal processing in the voice frequency band, thus enabling arithmetic operations at high speed and high accuracy to be done on LSIs. The HSP has a set of instructions that effectively carry out such floating point arithmetic operations, as well as fixed point arithmetic operations in general use in microcomputers.

(2) Horizontal-type microinstructions

To improve the throughput of arithmetic operations, the HSP offers horizontal-type microinstructions, so that more than one operation may be executed during a single instruction cycle.

A single instruction allows the following operations to be executed in parallel mode.

(a) ALU operation

(b) MULT operation

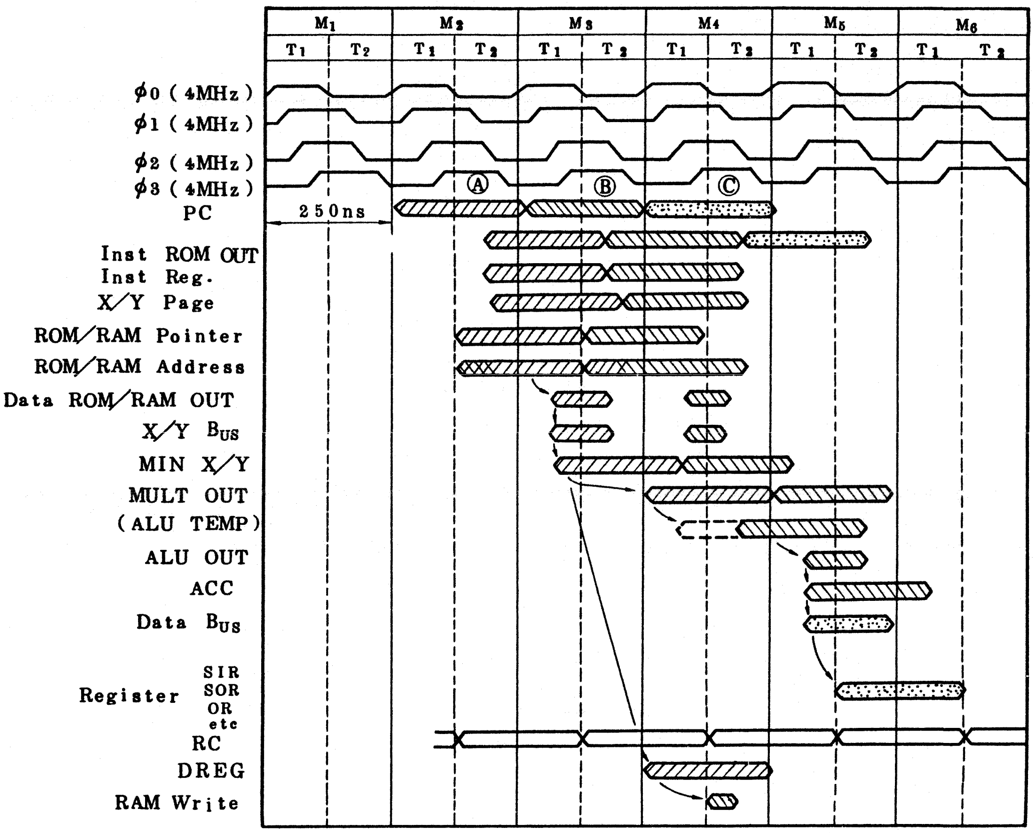
(c) Memory read

(d) Memory write

(e) Address pointer auto-increment and repeat counter auto-decrement

(3) Pipeline control

Computer pipeline control can give better throughputs to repeated product sum operations which are frequently seen in signal processing. The HSP cause multiply and add/subtract operations to be generated under pipeline control, so that the time required to execute a product sum operation is apparently equivalent to one instruction cycle. Pipeline control is also used for such operations as instruction prefetch and data memory read; this is enough to implement instructions of high throughput. The sequence of pipeline operations is shown in Fig. 7-1.



Ⓐ Multiply X1 Y1

Ⓑ Multiply X2 Y2

Add Acc + X1 Y1

Ⓒ Register transfer

Fig. 7-1 Pipeline Operation Sequence

7.2 Set of Instructions

Depending on their types, the HSP's instruction codes are classified as follows:

I. ALU operation
II. Immediate data
III. Jump
IV. Register transfer operation
V. Register increment/decrement
VI. Subroutine return

The HSP's instructions are configured so as to provide most effective operation for the instructions related to ALU operation.

This set of instructions includes no multiply-related instructions. Multiplication operates in every instruction cycle. If multiply operation is required, it is necessary to select input data of the multiplier and execute an ALU operation using the result of multiplication in the next cycle. In other words, it is necessary to select one of the addressing modes shown in Table 4-1 and thus set required addresses. Any instruction of the HSP, as shown in Fig. 7-2, consists of 32 bits.

In the following paragraphs are described the function of each instruction and the instruction formats for assembler description.

See Table 7-1, which lists the mnemonics, operations, instruction codes and CCR changes of the HSP's instructions.

The overflow protection for the result of arithmetic operation differs from one ALU operation instruction to another. To be more specific, the instructions may be divided into two groups. One is those instructions that placed under overflow protection even if the CTR's OVF is not set; and the other is those instructions that are under overflow protection only when the OVFP is set. This is detailed in Table 7-1.

Table 7-1 Instruction Formats

I	21 20 19 18 17	16 15	14	13	12 11	10	9	8 7 6	5 4 3	2	1	0
	OP			Address Control								
Pointer addressing mode	OP Code	(x, y) →ALU	FL/ FX	ACC A/B	ACC/ DREG Mem (Y)	1	X·Y / X·G	X-page	Y-page	RAM pointer INC.	ROM pointer INC.	RAM pointer A/B
										RC Dec		
Direct addressing mode						0	0	Direct address (X/Ypage)(Pointer address)				

Ⅱ	OP Code	ACC A/B — Immediate data (16 bits)	
		RC←Immediate data(6 bits)	
		ROM Pointer← Immediate data / RAM Pointer (6 bits)	

Ⅲ	OP Code	Jump Condition							Jump Address
		RC*							
		C	N	Z					

- ○ Jump if (Jump condition)∧(CCR)≠0
- ○ Jump condition =all"0" for noncoditional jump
- ○ When RC*=0, jump against (RC)≠0
- ○ When RC*=1, jump against (RC)≠0 and (RC)-1

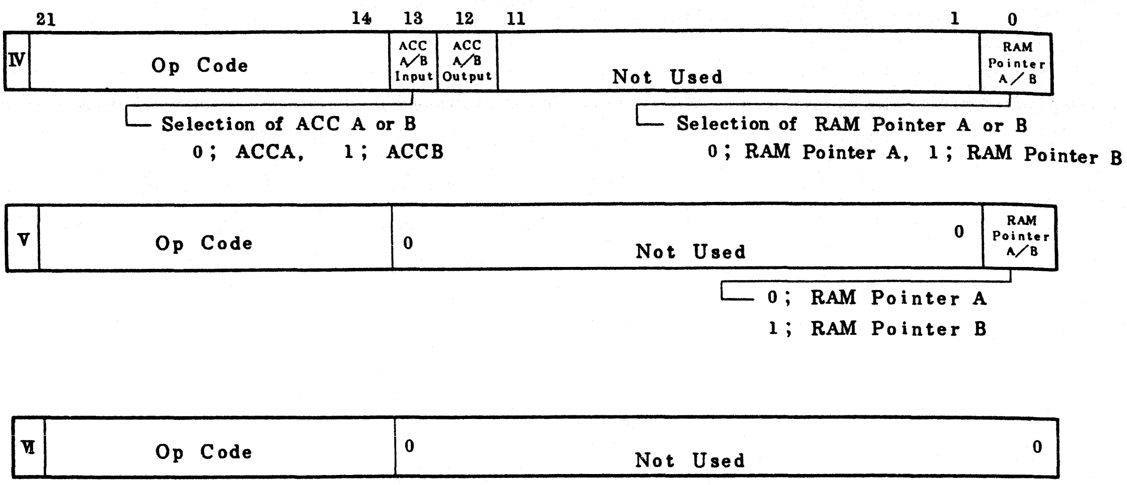


Fig. 7-2 HSP Instruction Formats

7.2.1 ALU Operation Instructions

Each instruction given here performs, at the same time, the following: ALU action, arithmetic operation, ALU input bus selection, data memory address control, read/write control, address pointer control, and repeat counter control. There are two addressing modes available.

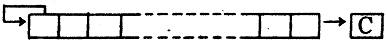
(1) ALU Operation

No	Mnemonic	Contents
1	FADA	Floating point arithmetic operation FL[(α)+(β) → ACCA]
2	FADB	Floating point arithmetic operation FL[(α)+(β) → ACCB]
3	ADA	Fixed point arithmetic operation FX[(α)+(β) → ACCA]
4	ADB	Fixed point arithmetic operation FX[(α)+(β) → ACCB]
5	FSBA	Floating point arithmetic operation FL[(α)-(β) → ACCA]
6	FSBB	Floating point arithmetic operation FL[(α)-(β) → ACCB]
7	SBA	Fixed point arithmetic operation FX[(α)-(β) → ACCA]
8	SBB	Fixed point arithmetic operation FX[(α)-(β) → ACCB]
9	FLDA	Floating point arithmetic operation FL[(β) → ACCA]
10	FLDB	Floating point arithmetic operation FL[(β) → ACCB]

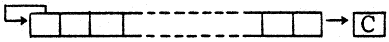
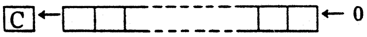
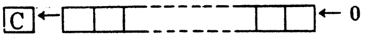
-cont'd-

No	Mnemonic	Contents
11	LDA	Fixed point arithmetic operation FX[$(\beta) \to$ ACCA]
12	LDB	Fixed point arithmetic operation FX[$(\beta) \to$ ACCB]
13	ANDA	Fixed point arithmetic operation FX[$(\alpha)\wedge(\beta) \to$ ACCA]
14	ANDB	Fixed point arithmetic operation FX[$(\alpha)\wedge(\beta) \to$ ACCB]
15	ORA	Fixed point arithmetic operation FX[$(\alpha)\vee(\beta) \to$ ACCA]
16	ORB	Fixed point arithmetic operation FX[$(\alpha)\vee(\beta) \to$ ACCB]
17	EORA	Fixed point arithmetic operation FX[$(\alpha)\oplus(\beta) \to$ ACCA]
18	EORB	Fixed point arithmetic operation FX[$(\alpha)\oplus(\beta) \to$ ACCB]
19	FABSA	FL[(no change in ACCA) → ACCA] where "no change in ACCA" means that there is no change in the absolute value of the mantissa and the exponent of the ACCA.
20	FABSB	FL((no change in ACCB) → ACCB] where "no change in ACCB" means that there is no change in the absolute value of the mantissa and the exponent of the ACCB.
21	ABSA	FX[(ACCA's absolute value) → ACCA]
22	ABSB	FX[(ACCB's absolute value) → ACCB]
23	FRPTA	Causes the next instruction to be repeated, where the operand is effective and multiplication is made in floating point form.

-cont'd-

No	Mnemonic	Contents
24	FRPTB	This causes the next instruction to be repeated, where the operand is effective and multiplication is made in floating point form.
25	RPTA	This causes the next instruction to be repeated, where the operand is effective and multiplication is made in fixed point form.
26	RPTB	This causes the next instruction to be repeated, where the operand is effective and multiplication is made in fixed point form.
27	FNEGA	FL[-(ACCA) → ACCA]
28	FNEGB	FL[-(ACCB) → ACCB]
29	NEGA	FX[-(ACCA) → ACCA]
30	NEGB	FX[-(ACCB) → ACCB]
31	INCA	FX[(ACCA)+1 → ACCA]
32	INCB	FX[(ACCB)+1 → ACCB]
33	DECA	FX[(ACCA)-1 → ACCA]
34	DECB	FX[(ACCB)-1 → ACCB]
35	SRA	This arithmetically shifts the ACCA to the right. 

-cont'd-

No	Mnemonic	Contents
36	SRB	This arithmetically shifts the ACCB to the right. 
37	SLA	This arithmetically shifts the ACCA to the left. 
38	SLB	This arithmetically shifts the ACCB to the left. 
39	FLTA	This converts the ACCA's mantissa part to floating point data (mantissa and exponent) and sets it in the ACCA. For details, see 3.2(3).
40	FLTB	This converts the ACCB's mantissa part to floating point data (mantissa and exponent) and sets it in the ACCB. For details, see 3.2(3).
41	FIXA	This converts the ACCA's floating point data (mantissa and exponent) to fixed point data (mantissa part) and sets it in the ACCA. For details, see 3.2(3).
42	FIXB	This converts the ACCB's floating point data (mantissa and exponent) to fixed point data (mantissa part) and sets it in the ACCB. For details, see 3.2(3).
43	FCLRA	FL[The contents of the ACCA are cleared.] (Mantissa = 0; exponent = -8)
44	FCLRB	FL[The contents of the ACCB are cleared.] (Mantissa = 0; exponent = -8)
45	CLRA	FX[The contents of the ACCA are cleared.]
46	CLRB	FX[The contents of the ACCB are cleared.]

-cont'd-

No	Mnemonic	Contents
47	FNOPA	FL [The ALU is a non-operation. The contents of the ACCA/B remain the same. The contents of the operand are effective.]
48	FNOPB	
49	NOPA	FX [The ALU is a non-operation. The contents of the ACCA/B remain the same. The contents of the operand are effective.]
50	NOPB	
51	FSGYB	FL [If the sign bit of data entered from the Y-bus is the same as the sign bit of the ACCA/B, the contents of the ACCA/B remain the same. If the sign bits are different, -(AACA/B) → ACCA/B]
52	FSGYB	
53	SGYA	FX [If the sign bit of data entered from the Y-bus is the same as the sign bit of the ACCA/B, the contents of the ACCA/B remain the same. If the sign bits are different, -(ACCA/B) → ACCA/B]
54	SGYB	

It should be noted that an interrupt operation must wait while a repeat instruction (FRPTA, FRPTB, BPTA or RPTB) is being executed or while any instruction which has been initiated by a repeat instruction is being cycled.

(2) Instruction Format

-- Page Addressing Mode (Type I)

Description in the assembly language looke like the following:

Label Δ Operation Δ [1]Δ[2]Δ[3] [4] ; Comment

where [1] - [4] are operands.

[1] Selection between ALU's two input data items (Note that this operand is not required for the instructions 19-54.)

Contents	Notation	Bit assignment	
		16	15
(product, ACC)→ALU	PA	0	0
(Y-Bus , ACC)→ALU	YA	0	1
(product, X-Bus)→ALU	PX	1	0
(Y-Bus , X-Bus)→ALU	YX	1	1

Note: In the contents block, ( ) refers to α and β in that order.

[2] RAM write control

Contents	Notation	Bit assignment	
		12	11
ACC→Data Bus, M(Y) not write	EE	0	0
ACC→Data Bus→M(Y) write	A	0	1
ACC→Data Bus, M(Y) not write	EE	1	0
ACC→Data Bus. DREG→M(Y) write	D	1	1

[3] Data memory output

Contents	Notation	Bit assignment			
		10	9	8 7 6	5 4 3
ROM/RAM→X or Y-Bus	XY(n, m)	1	0	n	m
ROM/RAM→X-Bus , GR→Y-Bus	XG(n, ℓ)	1	1	n	ℓ

(n,m)

n : Page address of output data to X-bus
m : Page address of output data to Y-bus
ℓ : GR address
0-3 : RAM
4-7 : ROM
0-3 : GR

[4] Automatically incrementing ROM/RAM pointer
Automatically incrementing repeat counter
Selecting RAM pointer

Contents		Notation	Bit assignment		
	RAM pointer selection		2	1	0
RAM Pointer ; not affected	A	RA, RO	0	0	0
ROM Pointer ; not affected	B	RB, RO	0	0	1
RAM Pointer ; auto increment	A	RA+, RO	1	0	0
ROM Pointer ; not affected	B	RB+, RO	1	0	1
RAM Pointer ; not affected	A	RA, RO+	0	1	0
ROM Pointer ; auto increment	B	RB, RO+	0	1	1
RAM Pointer ; auto increment	A	RA+, RO+	1	1	0
ROM Pointer ; auto increment	B	RB+, RO+	1	1	1

Automatically decrementing repeat counter (RC)

The logical sum (OR) of bits 2 and 1 is:

$2^2 \vee 2^1 = 0$; not affected

$2^2 \vee 2^1 = 1$; auto Decrement

(3) Instruction Format

-- Direct Addressing Mode (Type I)

Label Δ Operation Δ [1]Δ[2]Δ[3] ; Comment

where [1] - [3] are operands.

[1] Selecting ALU's two input data items

Same as with Page Addressing Mode.

[2] RAM write control

Same as with Page Addressing Mode.

[3] Direct addressing (n,m)

Contents	Bit assignment										
	10	9	8	7	6	5	4	3	2	1	0
ROM/RAM→Y − Bus (ACC→X − Bus	0	0	n			m					

n : Page address

m : Pointer address
(Note that memory is addressed directly, not by way of the pointer.)

(4) Multiplication

By addressing data memory with an ALU operation instruction, the data sent out to the X- or Y-bus is entered, as it is, into the multiplier. The multiplier operates whenever any instruction has been executed. The output (product) of the multiplier will be held only during the next instruction cycle. For the sum of products, it is necessary to select data to be sent out to the X- and Y buses for the instruction executed before the add instruction.

In the case of floating point product sum, specify successive arithmetic instructions in the form of floating point.

Example: (1) Correct description
FNOPA
FADA
(2) Incorrect description
FNOPA
ADA

7.2.2 Immediate Instructions

An immediate instruction sets the immediate data on the instruction ROM into the ACCA/B, RAM/ROM pointer, and RC.

(1) Operation

No	Mnemonic	Contents
1	LIA	Immediate data (16 bits) --► ACCA
2	LIB	Immediate data (16 bits) --► ACCB
3	LIRA	Immediate data (6 bits) --► RAM Pointer A
4	LIRB	Immediate data (6 bits) --► RAM Pointer B
5	LIRO	Immediate data (6 bits) --► ROM Pointer
6	LIRC	Immediate data (6 bits) --► Repeat Counter

(2) Instruction Format (Type II)

Label Δ Operation Δ [1] ; Comment

where [1] is an operand.

[1] Immediate data

Instructions	Immediate data bit assignment
LIA, LIB	15 ~ 0 (MSB) (LSB)
LIRA, LIRB, LIRO, LIRC	15 ~ 10 (MSB) (LSB)

7.2.3 Jump and Conditional Jump Instructions

There are some kinds of jump instructions available: unconditional jump, conditional jump, and subroutine jump. If the jump conditions are satisfied, then any interrupt operation must wait.

(1) Operation

No	Mnemonic	Contents
1	JCS	A jump occurs when the CCR's carry flag is a 1.
2	JNS	A jump occurs when the CCR's negative flag is a 1.
3	JZS	A jump occurs when the CCR's zero flag is a 1.
4	JSR	Subroutine jump (PC ---> stack 0, PC stack 0 ---> stack 1)
5	JNZ	A jump occurs when the repeat counter (RC) is not a 0.
6	JNZM	A jump occurs when the repeat counter (RC) is not a 0; at the same time, the repeat counter is decremented.
7	JMP	Unconditional jump

(2) Instruction Format (Types III & III')

Label Δ Operation Δ [1] ; Comment

where [1] is an operand.

[1] Jump address

This consists of nine bits: bit 8 through bit 0.

7.2.4 Data Transfer Instructions

A data transfer instruction is intended to transfer data between registers. Different registers have different data sizes and bit locations. Note that the contents of a register from which data is sent out remain the same.

Every data transfer instruction deals with fixed point data. After data transfer to the ACC, the value of the exponent part of the ACC selected is not guranteed.

(1) Operation

No	Mnemonic	Contents
1	TFR A, STR	Transfers the contents of bits 0-7 of ACCA to STR.
2	TFR B, STR	Transfers the contents of bits 0-7 of ACCB to STR.
3	TFR A, CTR	Transfers the contents of bits 0-7 of ACCA to CTR.
4	TFR B, CTR	Transfers the contents of bits 0-7 of ACCB to CTR.
5	TFR A, RC	Transfers the contents of bits 10-15 of ACCA to RC.
6	TFR B, RC	Transfers the contents of bits 10-15 of ACCB to RC.
7	TFR A, OR	Transfers the contents of bits 0-15 of ACCA to parallel output register OR.
8	TRF B, OR	Transfers the contents of bits 0-15 of ACCB to parallel output register OR.
9	TFR A, RO	Transfers the contents of bits 10-15 of ACCA to ROM pointer.

-cont'd-

No	Mnemonic	Contents
10	TFR B, RO	Transfers the contents of bits 10-15 of ACCB to ROM pointer.
11	TFR A, RA	Transfers the contents of bits 10-15 of ACCA to RAM pointer A.
12	TFR B, RA	Transfers the contents of bits 10-15 of ACCB to RAM pointer A.
13	TFR A, RB	Transfers the contents of bits 10-15 of ACCA to RAM pointer B.
14	TFR B, RB	Transfers the contents of bits 10-15 of ACCB to RAM pointer B.
15	TFR A, CCR	Transfers the contents of bits 13-15 of ACCA to CCR.
16	TFR B, CCR	Transfers the contents of bits 13-15 of ACCB to CCR.
17	TFR STR, A	Transfers the contents of STR to bits 0-7 of ACCA. After STR transfer, each of the PF, SIF and SOF flags is reset.
18	TFR STR, B	Transfers the contents of STR to bits 0-7 of ACCB. After STR transfer, each of the PF, SIF and SOF flags is reset.
19	TFR CTR, A	Transfers the contents of CTR to bits 0-7 of ACCA.
20	TFR CTR, B	Transfers the contents of CTR to bits 0-7 of ACCB.
21	TFR RC, A	Transfers the contents of RC to bits 10-15 of ACCA.
22	TFR RC, B	Transfers the contents of RC to bits 10-15 of ACCB.
23	TFR IR, A	Transfers the contents of parallel input register to bits 0-15 of ACCA.

-cont'd-

No	Mnemonic	Contents
24	TFR IR, B	Transfers the contents of parallel input register to bits 0-15 of ACCB.
25	TFR RO, A	Transfers the contents of ROM pointer to bits 10-15 of ACCA.
26	TFR RO, B	Transfers the contents of ROM pointer to bits 10-15 of ACCB.
27	TFR RA, A	Transfers the contents of RAM pointer A to bits 10-15 of ACCA.
28	TFR RB, A	Transfers the contents of RAM pointer B to bits 10-15 of ACCA.
29	TFR RA, B	Transfers the contents of RAM pointer A to bits 10-15 of ACCB.
30	TFR RB, B	Transfers the contents of RAM pointer B to bits 10-15 of ACCB.
31	TFR CCR, A	Transfers the contents of CCR to bits 13-15 of ACCA.
32	TFR CCR, B	Transfers the contents of CCR to bits 13-15 of ACCB.
33	TFR SIR, A	Transfers the contents (16 bits) of serial input register SIR to ACCA.
34	TFR SIR, B	Transfers the contents (16 bits) of serial input register SIR to ACCB.
35	TFR A, B	Transfers the contents of ACCA to ACCB. Only for fixed point data.
36	TFR B, A	Transfers the contents of ACCB to ACCA. Only for fixed point data.

(2) Instruction Format (Type IV)

```
Label Δ Operation Δ Reg 1, Reg 2 Δ ; Comment

where Reg 1 and Reg 2 are operands.
```

1) Operation : TFR for every instruction
2) Reg 1 : Register on the source side
3) Reg 2 : Register on the destination side

Refer to the mnemonic column.

7.2.5 Increment/Decrement Instructions

There are increment and decrement instructions for the address pointer and repeat counter.

(1) Operation

Mnemonic	Contents
INCRA	(RAM PointerA)+1→RAM PointerA
INCRB	(RAM PointerB)+1→RAM PointerB
INCRO	(ROM Pointer)+1→ROM Pointer
DECRA	(RAM PointerA)−1→RAM PointerA
DECRB	(RAM PointerB)−1→RAM PointerB
DECRO	(ROM Pointer)−1→ROM Pointer
DECRC	Repeat counter (RC) - 1→RC

(2) Instruction Format (Type V)

```
Label Δ Operation    ; Comment
```

At bit 0, only the RAM pointer selects between A and B. Note that assembler description is included in the mnemonic. (Bit 0 = 0/1 = RAM pointer A/B)

7.2.6 Subroutine Return Instructions

There are subroutine return and interrupt return instructions available. Any interrupt operation must wait while such an instruction is being executed.

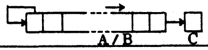
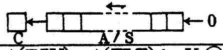
(1) Operation

No	Mnemonic	Contents
1	RTN	Subroutine return Stack 0 --→ PC Stack 1 --→ PC stack 0
2	RTI	Interrupt return Stack 0 --→ PC Stack 1 --→ PC stack 0 Resets the interrupt mask flag IM, enabling an interrupt.

(2) Instruction Format (Type VI)

Label Δ Operation ; Comment

Table 7-1 HSP Instruction Set

	MNEMONIC	OPERATION	INSTRUCTION																						OVF	CCR		
			21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		C	N	Z
I	FADA	P/Y+A/X→A(FLT)	0	0	0	0	0	0 ; ACC, 1 : X	0 ; P, 1 ; Y	0	0	0 ; A→M(Y), 1 ; D→M(Y)	Pointer addressing												1	①	↕	↕
	FADB	P/Y+B/X→B(FLT)	0	0	0	0	0			0	1														1	①	↕	↕
	ADA	P/Y+A/X→A	0	0	0	0	0			1	0														⑧	↕	↕	↕
	ADB	P/Y+B/X→B	0	0	0	0	0			1	1														⑧	↕	↕	↕
	FSBA	P/Y−A/X→A(FLT)	0	0	0	0	1			0	0														1	②	↕	↕
	FSBB	P/Y−B/X→B(FLT)	0	0	0	0	1			0	1														1	②	↕	↕
	SBA	P/Y−A/X→A	0	0	0	0	1			1	0			1	0;X·Y, ;X·G	X−Page			Y−Page			RAM Base 0; Not inc, 1; Inc	ROM Base 0; Not inc, 1; Inc	0 ; RA, 1 ; RB	⑧	↕	↕	↕
	SBB	P/Y−B/X→B	0	0	0	0	1			1	1														⑧	↕	↕	↕
	FLDA	P/Y→A (FLT)	0	1	1	0	0			0	0														*	0	↕	↕
	FLDB	P/Y→B (FLT)	0	1	1	0	0			0	1														*	0	↕	↕
	LDA	P/Y→A	0	1	1	0	0			1	0														*	0	↕	↕
	LDB	P/Y→B	0	1	1	0	0			1	1														*	0	↕	↕
	ANDA	P/Y∧A/X→A	0	0	1	1	0			1	0														*	③	↕	↕
	ANDB	P/Y∧B/X→B	0	0	1	1	0			1	1														*	③	↕	↕
	ORA	P/Y∨A/X→A	0	0	1	0	0	ALU		1	0														*	③	↕	↕
	ORB	P/Y∨B/X→B	0	0	1	0	0			1	1														*	③	↕	↕
	EORA	P/Y⊕A/X→A	0	0	1	0	1			1	0														*	③	↕	↕
	EORB	P/Y⊕B/X→B	0	0	1	0	1			1	1														*	③	↕	↕
I′	FABSA	|A|→A (FLT)	0	1	1	0	1	0	0	0	0			RC Inst 2v Inst 1=0 ; Not dec Inst 2v Inst 1=1 ; Dec											1	0	↕	↕
	FABSB	|B|→B (FLT)	0	1	1	0	1	0	0	0	1														1	0	↕	↕
	ABSA	|A|→A	0	1	1	0	1	0	0	1	0														⑧	0	↕	↕
	ABSB	|B|→B	0	1	1	0	1	0	0	1	1														⑧	0	↕	↕
	FRPTA	Repeat next (FLT)	1	1	1	1	0	0	0	0	0														*	●	●	●
	FRPTB	Instruction (FLT)	1	1	1	1	0	0	0	0	1														*	●	●	●
	RPTA	use A/B	1	1	1	1	0	0	0	1	0			Direct addressing											*	●	●	●
	RPTB	Until RC=0	1	1	1	1	0	0	0	1	1														*	●	●	●
	FNEGA	−A→A (FLT)	0	1	0	0	0	0	1	0	0			0	0	Page			Pointer						1	④	↕	↕
	FNEGB	−B→B (FLT)	0	1	0	0	0	0	1	0	1		0 ; Not write, 1 ; Write												1	④	↕	↕
	NEGA	−A→A	0	1	0	0	0	0	1	1	0														⑧	④	↕	↕
	NEGB	−B→B	0	1	0	0	0	0	1	1	1														⑧	④	↕	↕
	INCA	A+1→A Not protect	0	1	0	0	1	0	0	1	0														*	⑤	↕	↕
	INCB	B+1→B 〃	0	1	0	0	1	0	0	1	1														*	⑤	↕	↕
	DECA	A−1→A 〃	0	0	1	1	1	0	0	1	0														*	⑥	↕	↕
	DECB	B−1→B 〃	0	0	1	1	1	0	0	1	1														*	⑥	↕	↕
	SRA		0	1	0	1	0	0	0	1	0														*	↕	↕	↕
	SRB		0	1	0	1	0	0	0	1	1														*	↕	↕	↕
	SLA		0	1	0	1	1	0	0	1	0														*	↕	↕	↕
	SLB		0	1	0	1	1	0	0	1	1														*	↕	↕	↕
	FLTA	A(FIX)→A(FLT) by Y ⑨	0	1	1	1	1	0	1	0	0														*	0	↕	↕
	FLTB	B(FIX)→B(FLT) by Y ⑨	0	1	1	1	1	0	1	0	1														*	0	↕	↕
	FIXA	A(FLT)→A(FIX) by Y ⑩	0	0	0	1	0	0	1	0	0														*	0	↕	↕
	FIXB	B(FLT)→B(FIX) by Y ⑩	0	0	0	1	0	0	1	0	1														*	0	↕	↕
	FCLRA	00008→A	1	1	1	1	0	1	0	0	0														*	●	0	1
	FCLRB	00008→B	1	1	1	1	0	1	0	0	1														*	●	0	1
	CLRA	0000*→A	1	1	1	1	0	1	0	1	0														*	●	0	1
	CLRB	0000*→B	1	1	1	1	0	1	0	1	1														*	●	0	1
	FNOPA	ALL (FLT)	1	1	1	1	0	0	1	0	0														*	●	●	●
	FNOPB	no operation (FLT)	1	1	1	1	0	0	1	0	1														*	●	●	●
	NOPA	use A/B	1	1	1	1	0	0	1	1	0														*	●	●	●
	NOPB		1	1	1	1	0	0	1	1	1														*	●	●	●
	FSGYA	A/B→A/B (FLT)	0	1	1	1	0	0	1	0	0														1	⑦	↕	↕
	FSGYB	If sign A/B=sign Y (FLT)	0	1	1	1	0	0	1	0	1														1	⑦	↕	↕
	SGYA	−A/B→A/B	0	1	1	1	0	0	1	1	0														⑧	⑦	↕	↕
	SGYB	If sign A/B≠sign Y	0	1	1	1	0	0	1	1	1														⑧	⑦	↕	↕

-cont'd-

	OPERAND	OPERATION	INSTRUCTION																						OVF	CCR		
			21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		C	N	Z
Ⅱ	LIA	Immediate data→A	1	0	0	0	0	0	Data																*	●	↕	↕
	LIB	Immediate data→B	1	0	0	0	0	1																	*	●	↕	↕
	LIRA	Immediate data→RA	1	0	0	0	1	0	Data						0	0	0	0	0	0	0	0	0	0	*	●	●	●
	LIRB	Immediate data→RB	1	0	0	0	1	1							0	0	0	0	0	0	0	0	0	0	*	●	●	●
	LIRO	Immediate data→RO	1	0	0	1	0	0							0	0	0	0	0	0	0	0	0	0	*	●	●	●
	LIRC	Immediate data→RC	1	0	0	1	0	1							0	0	0	0	0	0	0	0	0	0	*	●	●	●
Ⅲ	JCS	Jump if C = 1	1	0	1	0	0	0	1	0	0	0	0	0	0	Jump address									*	●	●	●
	JNS	Jump if N = 1	1	0	1	0	0	0	0	1	0	0	0	0	0										*	●	●	●
	JZS	Jump if Z = 1	1	0	1	0	0	0	0	0	1	0	0	0	0										*	●	●	●
	JSR	Jump to subroutine	1	0	1	0	1	0	0	0	0	0	0	0	0										*	●	●	●
	JNE	Jump if RC≠0	1	0	1	0	1	1	0	0	0	0	0	0	0										*	●	●	●
	JNZM	Jump if RC≠0,RC−1→RC	1	0	1	0	1	1	1	0	0	0	0	0	0										*	●	●	●
	JMP	Jump always	1	0	1	0	0	0	0	0	0	0	0	0	0										*	●	●	●
Ⅳ (MNEMONIC: TFR)	A, STR	A→STR	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	B, STR	B→STR	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	A, CTR	A→CTR	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	B, CTR	B→CTR	1	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	A, RC	A→RC	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	B, RC	B→RC	1	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	A, OR	A→OR	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	B, OR	B→OR	1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	A, RO	A→RO	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	B, RO	B→RO	1	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	A, RA	A→RA	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	B, RA	B→RA	1	1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	A, RB	A→RB	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	*	●	●	●
	B, RB	B→RB	1	1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	*	●	●	●
	A, CCR	A→CCR	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	↕	↕	↕
	B, CCR	B→CCR	1	1	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	*	↕	↕	↕
	A, SOR	A→SOR	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	B, SOR	B→SOR	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	*	●	●	●
	STR, A	STR→A	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	STR, B	STR→B	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	CTR, A	CTR→A	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	CTR, B	CTR→B	1	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	RC, A	RC→A	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	RC, B	RC→B	1	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	IR, A	IR→A	1	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	IR, B	IR→B	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	RO, A	RO→A	1	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	RO, B	RO→B	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	RA, A	RA→A	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	RA, B	RA→B	1	1	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	RB, A	RB→A	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	*	●	↕	↕
	RB, B	RB→B	1	1	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	*	●	↕	↕
	CCR, A	CCR→A	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	CCR, B	CCR→B	1	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	SIR, A	SIR→A	1	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	SIR, B	SIR→B	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	A, B	A→B (FIX only)	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕
	B, A	B→A (FIX only)	1	1	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	*	●	↕	↕

-cont'd-

	MNEMONIC	OPERATION	INSTRUCTION																						OVF	CCR
			21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		C N Z
V	INCRA	RA+1→RA	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●●●
	INCRB	RB+1→RB	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	*	●●●
	INCRO	RO+1→RO	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●●●
	DECRA	RA−1→RA	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●●●
	DECRB	RB−1→RB	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	*	●●●
	DECRO	RO−1→RO	1	1	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●●●
	DECRC	RC−1→RC	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●●●
Ⅵ	RTI	Returm from interrupt	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●●●
	RTN	Returm from subroutine	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	●●●

A	Accumulator A
B	Accumulator B
X	X−BUS memory output
Y	Y−BUS memory output
P	Multiplier output register
G	General register
D	Delay register
PC	Program counter
RC	Repeat counter
RO	ROM Pointer
RA	RAM Pointer A
RB	RAM Pointer B
IR	Input register
OR	Output register
SIR	Serial input register
SOR	Serial output register

CCR	Condition code register
STR	Status register
CTR	Control register

A/B(15∼10)↔RC/RO/RA/RB(5∼0)
A/B(15∼13)↔CCR(2∼0)
A/B(7∼0)↔STR/CTR(7∼0)

	7	6	5	4	3	2	1	0
CCR						C	N	Z
CTR	W/B	/	TxRQF	DIT IOF	/	/	OVF	DMA
STR	UF	Iso	IsI	IP	IM	SOF	SIF	PF

① Generated by the mantissa part of two inputs
(The same as with FIX)

② Generated by the mantissa part of two inputs
(The same as with FIX)

③ Indefinite

④ 1 when A/B = $0000; otherwise 0

⑤ 1 when A/B = $FFFF; otherwise 0

⑥ 1 when A/B ≠ $0000; otherwise 0

⑦ 1 when sign A/B ≠ sign Y and A/B = $0000; otherwise 0

⑧ OVF (CTR(1)) = 0 : Not protect, 1 : Protect

⑨ Y, Mant : free

⑩ Y, Mant : $0000, Automatic Overflow Protect

↕ Affected

● Not affected

* free for CTR (1)

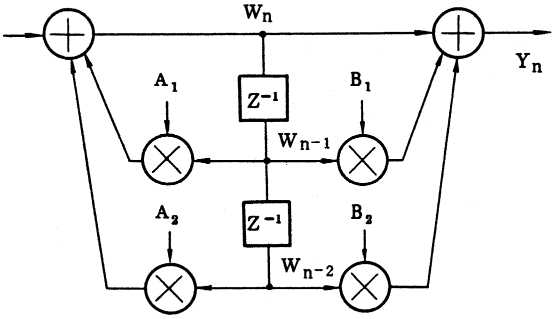
In the OVFP column, a "1" means "Automatic Overflow Protect".

7.3 Examples of Program

Here are examples of program for the typical filters: biquad filter and transversal filter. See Fig. 7-3 and Fig. 7-4.

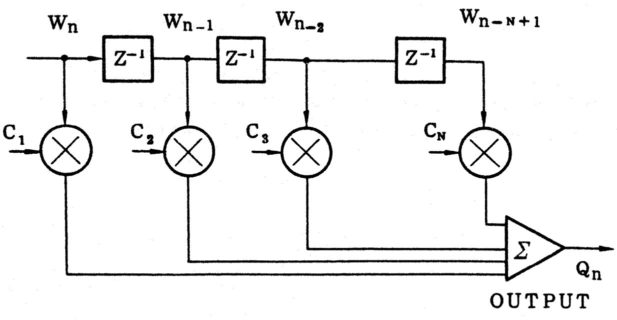
The coding of the program is based on the HSP's assembler description.

The example of the biquad filter follows the sequence: receives 16-bit data from the serial input register (SIR), makes an arithmetic operation on it using fixed point representation, and sets the result in the serial output register (SOR). One stage of processing is contained in the filter's arithmetic operation. Since the HSP has an instruction cycle of 250ns, the rate of processing for 8kHz sampling is great enough to implement a 82-stage biquad filter. Because the data ROM for coefficients is limited to 128 words, however, it is practically possible to implement a 32-stage biquad filter.



$$W_n = X_n + A_1 * W_{n-1} + A_2 \times W_{n-2}$$
$$Y_n = W_n + B_1 * W_{n-1} + B_2 \times W_{n-2}$$
$$W_{n-1} \rightarrow W_{n-2},\ W_n \rightarrow W_{n-1}$$

Fig. 7-3 Biquad Filter



$$Qn=\sum_{i=1}^{N} Ci*Wn-i+1$$

$$Wn-i+1 \rightarrow Wn-i,\ N=32$$

Fig. 7-4 Transversal Filter

In the example of the transversal filter, the sequence is: receives data from the serial input register (SIR), converts the data from fixed point to floating point, makes a transversal filter arithmetic operation in the form of flaoting point, converts the data again from floating point to fixed point, and finally sends out to the serial output register (SOR).

This program enables a 32-tap transversal filter to be executed in a period of 10.5 microseconds.

(1) Biquad Filter

A biquad filter can be represented as

$$W_n = X_n + A_1 * W_{n-1} + A_2 * W_{n-2} \qquad (1)$$

$$Y_n = W_n + B_1 * W_{n-1} + B_2 * W_{n-2} \qquad (2)$$

The assembler program for these equations is

```
       LIRA          0         ; 0 → RAM Pointer
       LIRO          0         ; 0 → ROM Pointer
       TFR         SIR, A      ;SIR→ ACCA
       LIRC          n         ; n → RC
L1     NOPA EE  XY(4, 0) RA    ; A1 *Wn-1
```

```
ADA    PA EE  XY(5, 1)  RA      ; A2*Wn-2   ACCA+P → ACCA

ADA    PA EE  XY(7, 1)  RA      ; B2*Wn-2'  ACCA+P → ACCA

ADA    PA A   XY(6, 0)  RA      ; B1*Wn-1'  ACCA+P → ACCA,

                                  Wn-1 → DREG.,  ACCA → Wn-1

ADA    PA D   XY(7, 1)  RA+, RO+    ; DREG. → Wn-2'  ACCA+P → ACCA,

                                      RAM POINTER + 1, ROM POINTER + 1
JNZ           L1                ; Jump to L1 if(RC)≠0
TFR           A, SOR            ; ACCA → SOR
```

Data memory		Page address							
		0	1	2	3	4	5	6	7
Pointer address	00	Wn-1	Wn-2			A_1	A_2	B_1	B_2
	01	Wn'-1	Wn'-2			A_1'	A_2'	B_1'	B_2'
	02	Wn''-1	Wn''-2			A_1''	A_2''	B_1''	B_2''
	03								

(2) Transversal Filter

The 32-tap transversal filter is represented by;

$$Q = \sum_{i-1}^{32} C_i * W_{n-i+1} \qquad (3)$$

$$W_{n-i+1} \rightarrow W_{n-i} \qquad (4)$$

One sampling delay of the filtering is shown as Eq.(7). The assembler program for these equations is

```
LIRO         0              ; 0 → ROM POINTER
LIRA         0              ; 0 → RAM POINTER
LIRC         31             ; 31 → RC
TFR          SIR, A         ;SIR→ ACCA
FLTA         EE 6, 00       ;Fixed → Floating
FCLRA        A 0, 00        ; 0 → ACCA, ACCA → Wn
FRPTA        EE 4, 00       ;Next inst. repeat
FADA  PA D XY(4, 0) PA+, RO+ ; ACCA+C i*Wn-i+1,  Wn-i+1 → Wn-i

FADA      PA EE 0, 00       ;ACCA+C32 *Wn-31

FIXA         EE 6, 00       ;Floating → Fixed
TFR          A, SOR         ;ACCA → SOR
```

10.5μs

Data memory		Page address							
		0	1	2	3	4	5	6	7
Pointer address	00	Wn				C_1			
		Wn -1				C_2			
		Wn -2				C_3			
		⋮				⋮			
	31	Wi -n+1				Cn			

8. APPLICATION SYSTEM CONFIGURATION

8.1 Stand-alone Configuration

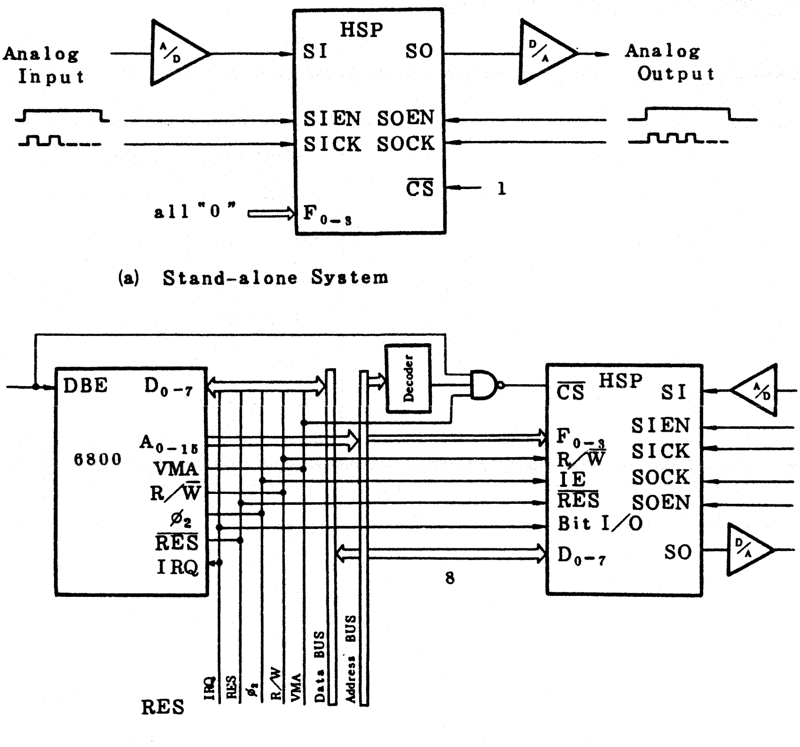
The HSP has functions and instructions as possessed by a single-chip microcomputer, so that it is possible to configure a system consisting of one or more HSP units. See Fig. 8.1(a).

By making the chip select input terminal $\overline{CS}$ non-active ("1"), this configuration disables HSP control based on function control input F0-F3. The result is that the HSP operates only in the internal program mode.

8.2 Peripheral LSI of 8-Bit Microcomputer (6800)

Shown in Fig. 8.1(b) is the interface where the HSP is used as the peripheral LSI of an 8-bit microcomputer (6800).

Because of being compatible with the 6800 bus interface, the HSP requires no complicated interface, as understood from Fig. 8.1(b). The bit I/O output is of open drain output and so may be used also as interrupt signals to the microcomputer.



(a) Stand-alone System

(b) μ—Com (6800) Interface

Fig. 8-1 HSP's Application System Configuration

9. ELECTRICAL CHARACTERISTICS

9.1 Absolute Maximum Rating

Parameter	Symbol	Standard range	Unit
Power voltage	V_{CC}	-0.3 to +7.0	V
Terminal voltage	V_{in}	-0.3 to V_{CC}+0.3	V
Operating temperature	T_{opr}	-20 to +70	°C
Storage temperature	T_{stg}	-55 to +150	°C

9.2 Electrical Characteristics

(1) DC Characteristics

Unless otherwise specified, V_{CC} = 5.0V ±5%, V_{SS} = 0V and T_a = -20 to +70°C.

Parameter		Symbol	Condition	Min	Typ	Max	Unit
Input HIGH level voltage	OSC, IE, SICK, SOCK	V_{IH}		2.2	-	V_{CC}+0.3	V
	Other input terminals			2.0	-	V_{CC}+0.3	V
Input LOW level voltage	OSC, IE, SICK, SOCK	V_{IL}		-0.3	-	0.6	V
	Other input terminals			-0.3	-	0.8	V
Input leak current	TEST, $\overline{\text{TXAK}}$, IE, R/$\overline{\text{W}}$, $\overline{\text{CS}}$, F0-F3, DEND, SI, SIEN, SOCK, SOEN, SICK, RPSE, OSC	$\|I_{IN}\|$	V_{in}= 0 to 2.4V	-	-	10	μA
Three-state (off) current	D0-D15, SO	$\|I_{TSI}\|$	V_{in}= 0 to 2.4V	-	-	10	μA
Open drain (off) current	TxRQ, BIT I/O	$\|I_{LOH}\|$	V_{in}= 0 to 2.4V	-	-	10	μA

-cont'd-

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Output HIGH level voltage D0-D15, SO, SYNC	V_{OH}	$-I_{OH}$= 400μA	2.4	-	-	V
		$-I_{OH}$= 10μA	V_{CC}-0.5	-	-	V
Output LOW level voltage All output terminals	V_{OL}	I_{OL}= 1.6mA	-	-	0.4	V
Input capacity All input terminals	C_{in}	V_{in}= 0V, f = 1MHz, T_a = 25°C	-	-	12.5	pF
Current consumption	I_{CC}	No load at output	-	50	100	μA

(2) AC Characteristics (Basic Clock)

Unless otherwise specified, V_{CC}= 5.0V ±5%, V_{SS}= 0V and T_a= -20 to +70°C.

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Clock (OSC) cycle	$Ø_{cyc}$	See Fig. 9-1.	61.5	62.5	70.0	ns
Clock (OSC) pulse width	$Ø_{WH}$		20	-	-	ns
	$Ø_{WL}$		20	-	-	ns
Clock (OSC) rise time	$Ø_r$		-	-	10	ns
Clock (OSC) fall time	$Ø_f$		-	-	10	ns

(3) Serial Input/Output Timing

Unless otherwise specified, V_{CC}= 5.0V ±5%, V_{SS}= 0V and T_a= -20 to +70°C.

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Clock cycle (SICK/SOCK)	S_{cyc}	See Figs. 9-2 and 9-5.	1.0	-	10.0	µs
Clock pulse width (SICK/SOCK)	S_{WH}		450	-	-	ns
	S_{WL}		450	-	-	ns
Clock rise time (SICK/SOCK)	S_r		-	-	25	ns
Clock fall time (SICK/SOCK)	S_f		-	-	25	ns
Serial input data setup time	t_{SDS}		100	-	-	ns
Serial input data hold time	t_{SDH}		100	-	-	ns
Serial output data delay time	t_{SDD}		-	-	250	ns
Enable delay time	t_{ED}		50	-	-	ns
Enable setup time	t_{ES}		100	-	-	ns

(4) Bus Interface Timing

Unless otherwise specified, V_{CC}= 5.0V ±5%, V_{SS}= 0V and T_a= -20 to +70°C.

Parameter	Symbol	Condition	Min	Typ	Max	Unit
IE cycle	t_{cyc}	See Figs. 9-3 and 9-5.	1.0	-	10.0	μs
IE pulse width	t_{WH}		450	-	-	ns
	t_{WL}		450	-	-	ns
IE rise time	t_r		-	-	25	ns
IE fall time	t_f		-	-	25	ns
CS setup time	t_{CS}		140	-	-	ns
CS hold time	t_{CH}		10	-	-	ns
Address setup time	t_{AS}		140	-	-	ns
Address hold time	t_{AH}		10	-	-	ns
Address setup time	t_{AC}		10	-	-	ns
Address hold time	t_{CA}		20	-	-	ns
Input data setup time	t_{DSW}		120	-	-	ns
Input data hold time	t_{DHW}		10	-	-	ns
Output data delay time	t_{DDR}		-	-	220	ns
Output data hold time	t_{DHR}		-	-	10	ns

(5) DMA Interface Timing

Unless otherwise specified, V_{CC}= 5.0V ±5%, V_{SS}= 0V and T_a= -20 to +70°C.

Parameter	Symbol	Condition	Min	Typ	Max	Unit
TxAK setup time	t_{AS}	See Figs. 9-4 and 9-5.	140	-	-	ns
TxRQ delay time	t_{TR}		-	-	470	ns

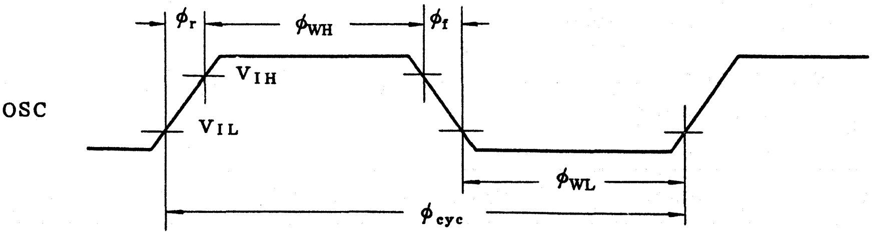


Fig. 9-1 Basic Clock Waveform

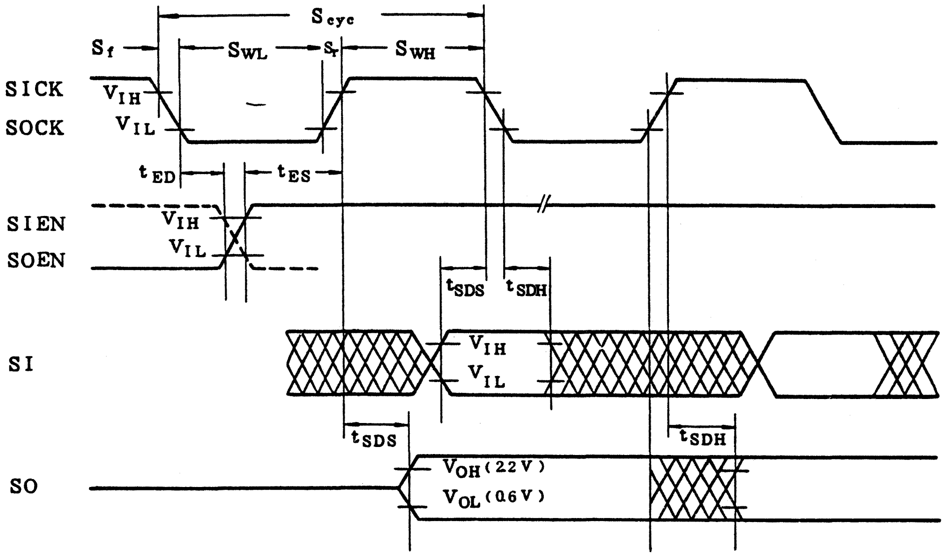
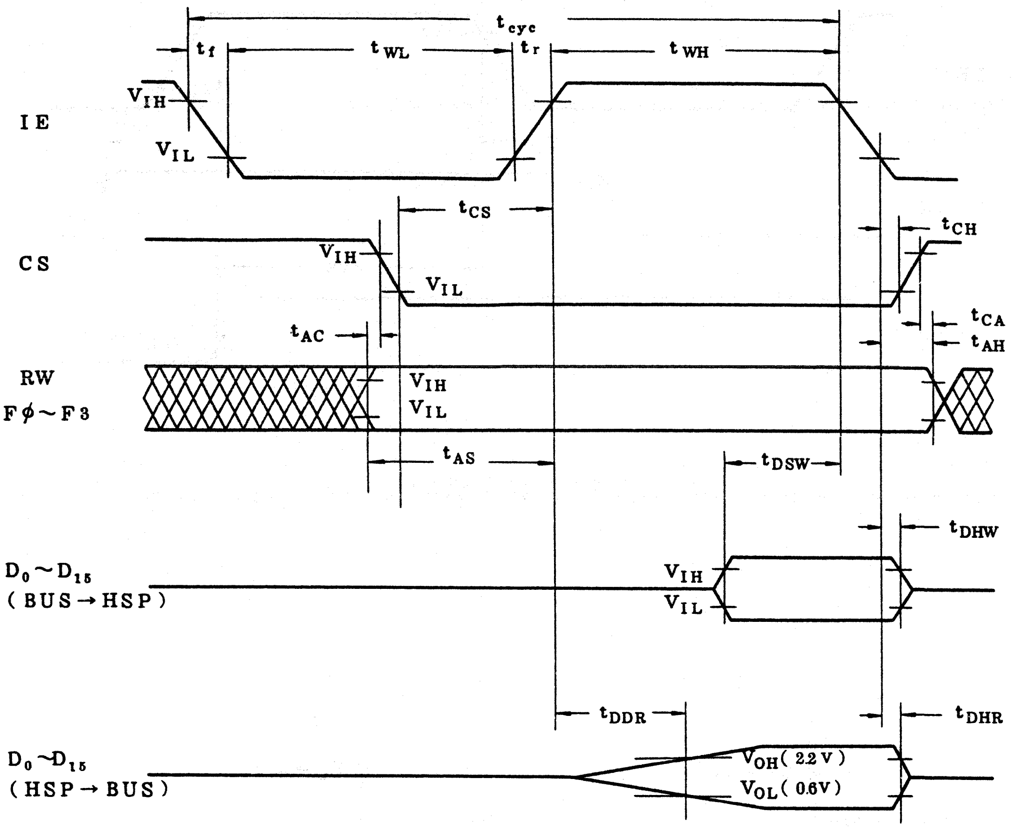


Fig. 9-2 Serial Input/Output Waveform



Note: The HSP makes a halt operation at BUS --→ CTR, PC. The halt consists of the logic of $\overline{CS}$ and F0-F3. It is necessary to secure the above t_{AC} and t_{CH} values to prevent a temporary halt at a point where the signal changes. If, however, no halt is expected to take place at such a change (F2 and F3 are fixed at LOW), the t_{AC} and t_{CA} conditions are each "don't care".

Fig. 9-3 Bus Timing

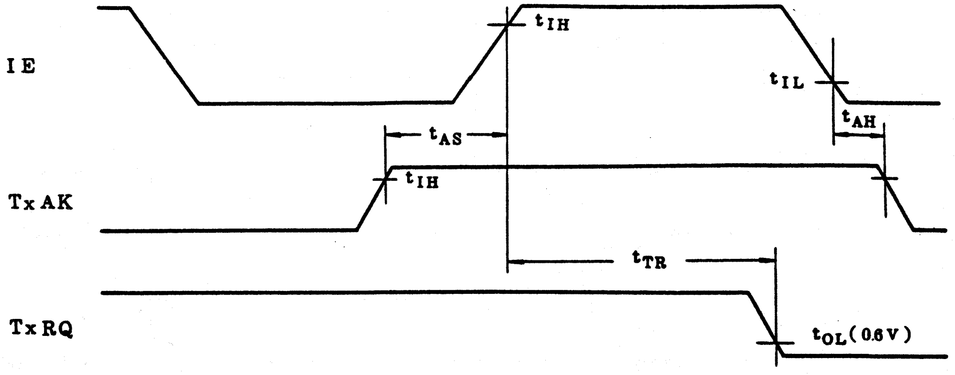
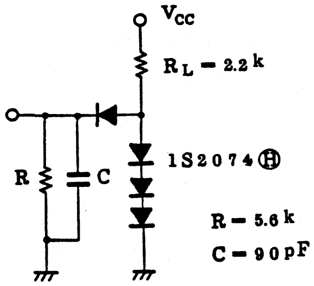
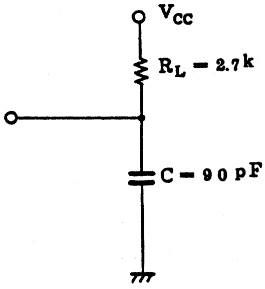


Fig. 9-4 DMA Timing





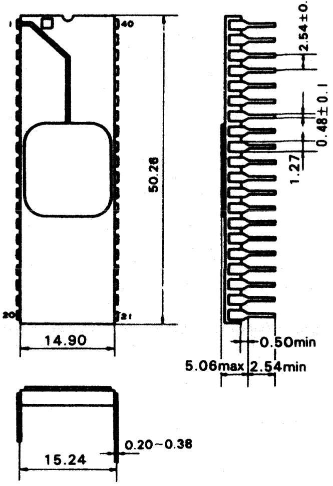
Applicable terminal
(D0 - D15, SO)

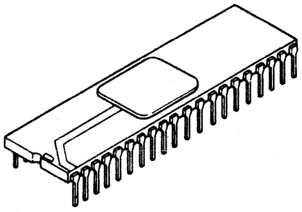
Applicable terminal
(Tx RQ)

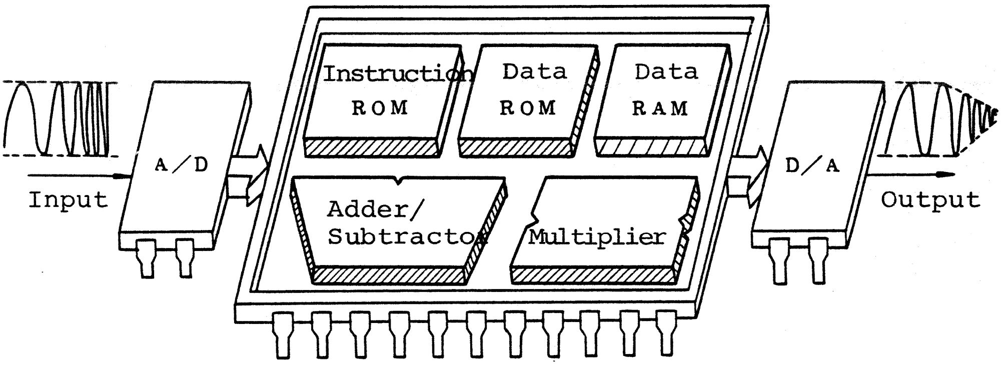
Fig. 9-5 Load Circuit (for Timing Test)

10. EXTERNAL APPEARANCE AND DIMENSIONS

(DC-40)






Instruction ROM
Data ROM
Data RAM
A/D
Input
Adder/ Subtractor
Multiplier
D/A
Output