NEC
NEC Electronics Inc.

μPD77017
16-Bit Fixed Point Digital Signal Processor

November 1994

Introduction

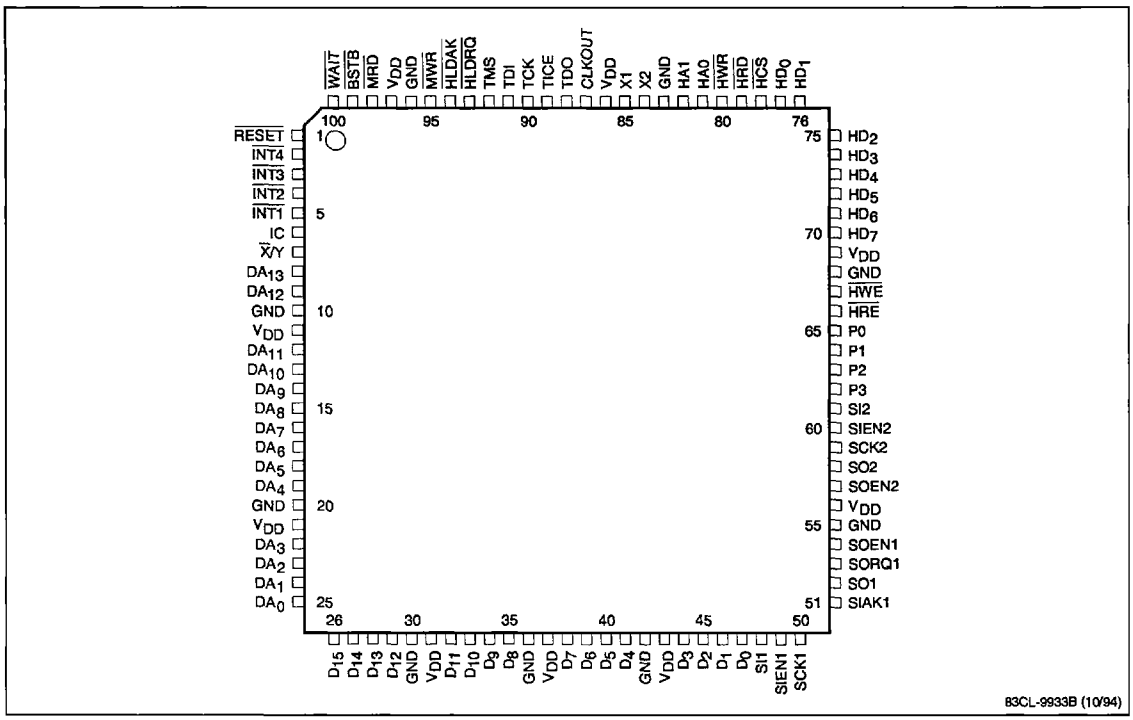
The μPD77017 is the second in NEC's SPRX family of high-speed 16-bit digital signal processors (DSPs). Low power consumption, a 3-volt power requirement, and two standby modes that reduce battery consumption make the μPD77017 ideal for hand-held, portable applications. A 12K-word instruction ROM accommodates the voice and data compression/decompression methods used in North America, Europe and Japan. Design support is provided by a comprehensive set of easy-to-use Windows™-based development tools, including a C compiler, for program writing, debugging, and testing. The μPD77017 is packaged in a low-profile, 100-pin thin quad flat pack (TQFP).

Features

- 0.5-micron process technology
- 30 ns (min) instruction cycle (33 MHz clock)
- Low power consumption: 1.21 mA/MIPS
- Power down mode: 6 μW and 9 mW
- Dual load/store capability
- Nested zero-overhead loops
- Conditional execution of many operations
- Ten interrupt sources: 4 external and 6 internal
- On-chip oscillator uses 33 MHz crystal
- On-chip instruction ROM = 12K words x 32 bits
- On-chip data ROM = 8K words x 16 bits
- On-chip data RAM = 4K words x 16 bits
- External data memory = 32K words x 16 bits
- 16 bits x 16 bits + 40 bits → 40-bit multiply accumulator
- Eight 40-bit general registers—fully orthogonal
- Four-bit programmable I/O port
- Two-channel, 16-bit serial interface
- One 8-bit external parallel host interface
- Single 3-volt power supply (2.7 to 3.6 volts)
- 100-pin TQFP (14 mm x 14 mm x 0.5 mm)

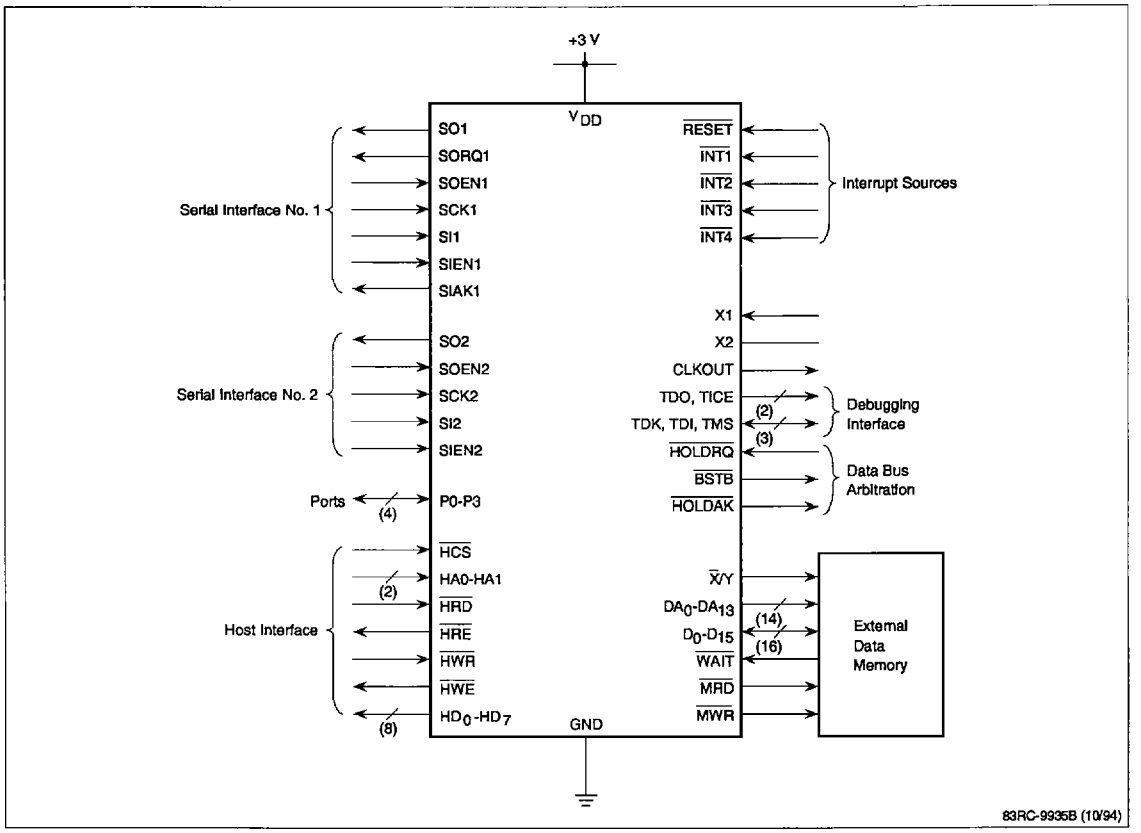
50669-2

Pin Configuration



83CL-9933B (10/94)

Functional Pin Groups



Functional Differences Among the μPD77015 , 77016, 77017, and 77018

Item	μPD77016	μPD77015	μPD77017	μPD77018
Internal instruction RAM	1.5K words	256 words	256 words	256 words
Internal instruction ROM	None	4K words	12K words	24K words
External instruction memory	48K words	None	None	None
Data RAM (X/Y memory)	2K words each	1K words each	2K words each	3K words each
Data ROM (X/Y)	None	2K words each	4K words each	12K words each
External data memory (X/Y memory)	48K words each	16K words each	16K words each	16K words each
Clock (operation clock = 33 MHz)	66 MHz	33, 16.5. 8.25, 4.125, 2.0625 MHz Variable multiple rates (1, 2, 4, 8, 16) by mask option		
Crystal (operation clock = 33 MHz)	—	33 MHz	33 MHz	33 MHz
Instruction	—	STOP instruction is added.		
Serial interface (2 channels)	Channel 1 has the same functions as channel 2	Channel 1 has the same functions as the μPD77016. Channel 2 has no SORQ2 and SIAK2 pins (Channel 2 is used for CODEC connection)		
Power supply	5 volts	3 volts	3 volts	3 volts
Package	160-pin plastic PQFP	100-pin plasticTQFP	100-pin plasticTQFP	100-pin plasticTQFP

PIN FUNCTIONS

Table 1 describes the μPD77017 pin functions on the 100-pin thin plastic QFP package. Pins are listed in alphabetical order by symbol with power pins listed at the end.

Table 1. Pin Functions

Symbol	I/O	Function
$\overline{BSTB}$	Out	Bus strobe output; low level while the μPD77017 owns external memory bus
CLKOUT	Out	Clock output
D_0 - D_{15}	I/O (3S)	16-bit data to external data memory (Notes 1, 4)
DA_0 - DA_{13}	Out (3S)	External data memory address (Notes 1, 4)
HA1	In	Specifies which register accessed by HD_0 - HD_7 1 = Host Interface Status Register (HIS) 0 = Host Transmit Data Register ($\overline{HRD}$ = 0) 0 = Host Receive Data Register ($\overline{HWR}$ = 0)
HA0	In	Specifies register bits accessed by HD_0 - HD_7 1 = Accesses bits 8-15 of HIS, HDT, or HDR 0 = Accesses bits 0-7 of HIS, HDT, or HDR
$\overline{HCS}$	In	Chip select input signal
HD_0 - HD_7	I/O (3S)	8-bit host data bus (Notes 3, 4)
$\overline{HOLDAK}$	Out	Hold acknowledge signal. The 77017 pulls this signal low to release the bus to an external bus master that has requested it.
$\overline{HOLDRQ}$	In	Hold request input; external bus master pulls low to request the bus
$\overline{HRD}$	In	Host read input signal
$\overline{HRE}$	Out	Host read enable output signal
$\overline{HWE}$	Out	Host write enable output signal
$\overline{HWR}$	In	Host write input signal
$\overline{INT1}$	In	Maskable external interrupt that is detected at the falling edge
$\overline{INT2}$	In	Maskable external interrupt that is detected at the falling edge
$\overline{INT3}$	In	Maskable external interrupt that is detected at the falling edge
$\overline{INT4}$	In	Maskable external interrupt that is detected at the falling edge
$\overline{MRD}$	Out (3S)	Active for a read of external data memory (Notes 1, 4)
P_0	I/O	General input/output port
P_1	I/O	General input/output port
P_2	I/O	General input/output port
P_3	I/O	General input/output port
$\overline{MWR}$	Out (3S)	Active for a write of external data memory (Notes 1, 4)
$\overline{RESET}$	In	Reset input
SCK1	In	Serial 1 clock input signal
SORQ1	Out	Serial output 1 request signal
SOEN1	In	Serial-1 output enable signal
SO1	Out (3S)	Serial-1 data output (Notes 3, 4)
SIEN1	Out	Serial-1 input enable signal
SIAK1	Out	Serial-1 input acknowledge signal
SI1	In	Serial-1 data input
SCK2	In	Serial-2 clock input signal
SOEN2	In	Serial-2 output enable signal
SO2	Out (3S)	Serial-2 data output (Notes 3, 4)
SIEN2	Out	Serial-2 input enable signal
SI2	In	Serial-2 data input
TDO	Out	Signal for debugging
TICE	Out	Signal for debugging
TCK	In	Signal for debugging
TDI	In	Signal for debugging
TMS	In	Signal for debugging
$\overline{WAIT}$	In	Used to extend an external memory read operation 1 = wait 0 = no wait
X1	In	Clock input/crystal connection pin
X2	—	Crystal connection; should be left open when using external clock for system clock
$\overline{X/Y}$	Out (3S)	Memory select (Notes 3, 4) 0 = X memory 1 = Y memory
V_{DD}	—	+3 V
GND	—	Ground
I.C.		Internally connected pin. Leave this pin open. Caution: When a signal is applied to or read from this pin, normal operation of the μPD77017 is not assured.

Notes:

(1) These signals are in a high-impedance state when the external data bus control is relinquished (HLDAK = LOW).

(2) These signals are in a high-impedance state when hardware reset ($\overline{RESET}$ = 0) is asserted.

(3) These signals are in high-impedance state when hardware reset ($\overline{RESET}$ = 0) is asserted or when data transfer throught serial interface is complete.

(4) The notation 3S indicates that the output pin may be tri-stated.

FUNCTIONAL OPERATION

The μPD77017 is a high-performance, 16-bit fixed-point DSP. With its eight 40-bit general purpose registers, the device efficiently executes DSP algorithms by exploiting separate data and instruction memory spaces. Data and instructional memory spaces are accessed via separate data buses. The device contains 12K x 32 bits of mask-programmable instruction ROM and two banks of internal data memory, each consisting of a 2K x 16-bit data RAM and 4K x 16-bit data ROM. In addition, there are 256 words x 32 bits of instruction RAM to support interrupt handlers and program patches. The data memory spaces can each be extended externally to 16 bits. Instructions and data are accessed via separate external buses. The μPD77017 block diagram shows the device's main features, which include:

- Multiple data buses
- Register-based Execution Unit (REU)
- Data Address Control Unit (DACU)
- Program Control Unit (PCU)
- Data and instruction memory
- Peripheral Unit (PU)

Multiple Data Buses

The μPD77017 incorporates three internal 16-bit data buses: the X Bus, the Y Bus and the Transfer Bus. The X and Y Buses move data between general registers and the X and Y data memory areas, respectively. The Transfer Bus moves data between general and control registers.

Block Diagram

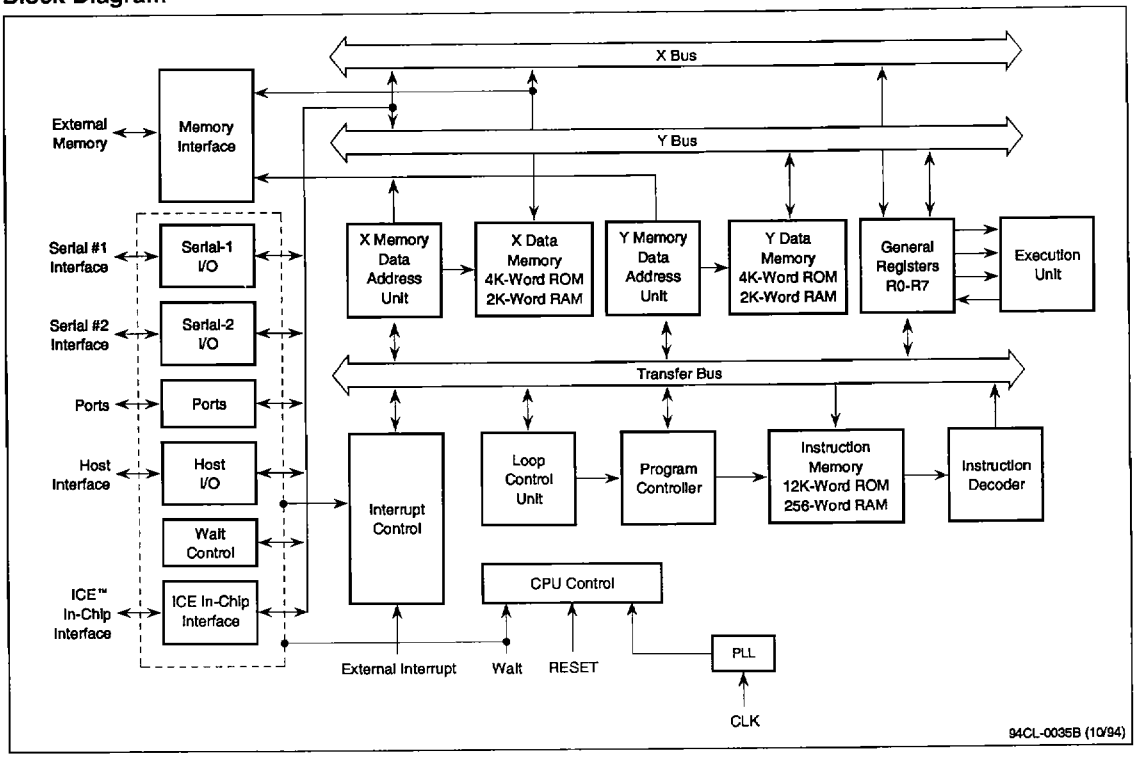
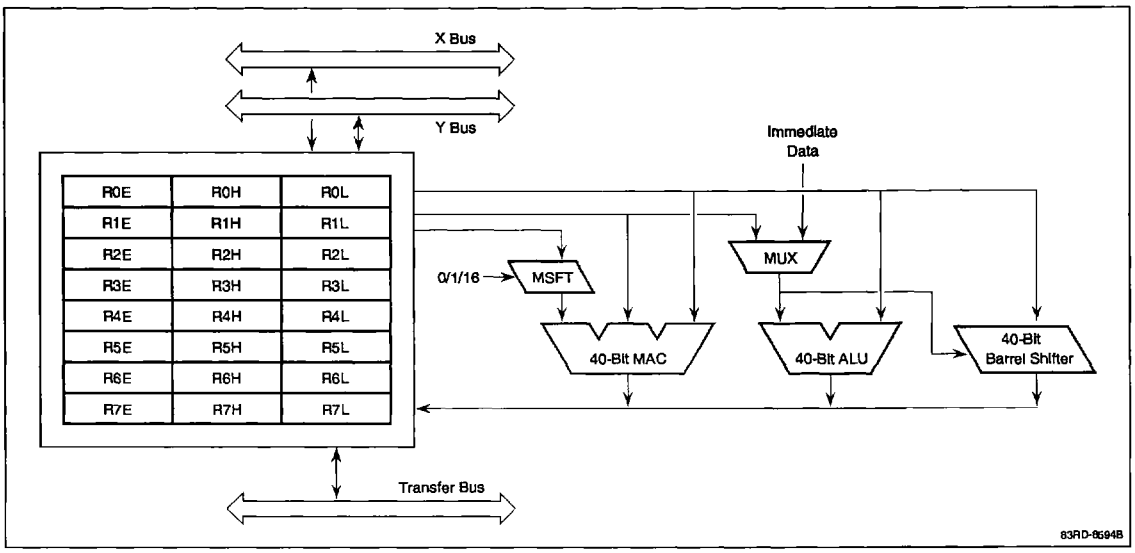


Figure 1. Execution Unit



Register-Based Execution Unit (REU)

The execution unit, illustrated in figure 1, shows the device's 40-bit multiply-accumulator (MAC), 40-bit data ALU and 40-bit barrel shifter and Shift-and-Count circuit. The 40-bit execution unit permits the accumulation of many 16-bit products without loss of accuracy, and it reduces the difficulty associated with overflow checking and correction.

General Registers (R0 - R7). These eight 40-bit general registers are available as operands for all arithmetic and logical operations of the μPD77017. Data transfer operations provide access to memory and to other registers.

Each general register is divided into three segments (denoted by the segment suffix L, H, or E). The three segments are:

RnL	Low 16-bit word (bits 0-15)
RnH	High 16-bit word (bits 16-31)
RnE	Extension (bits 32-40)

Each segment can be treated as an individual register under certain circumstances. The RnL segment is often used for integer values and the RnH for fractional data values. The sign of data is automatically extended into RnE when 16-bit data is written a general register from data memory, but it is also possible to load E, H or L segments without sign extension. General registers can be treated as either 40-bit, 32-bit, 24-bit, 16-bit, or 8-bit registers depending on the operation.

40-Bit General Registers. A general register with no segment suffix is considered to be 40-bits wide when it is used as:

- An input operand for a three operand operation (excluding input to MAC)
- An input operand for a two operand operation (excluding input to MAC or shift amount)
- An input operand to a immediate operand operation (excluding exponent operation)
- An output operand for the execution unit
- An input operand for a test condition

32-Bit General Registers. A general register with no segment suffix is treated as 32-bits wide (bits 0-31) only when it is used as the input operand for the exponent operation. If the register specified as the source for this operation contains a value above 0x80000000 or less than 0xFF7FFFFFFFF, the high order eight bits are ignored.

24-Bit General Registers. A general register with the suffix EH is treated as a 24-bit register when it is used as the destination register for a load operation. In this case, bits 24-39 are either set or cleared to effect sign extension, but bits 0-15 are ignored.

16-Bit General Registers. Bits 16-31 of a general register (H-suffix specified) are treated as a 16-bit register when used as:

- An input operand for MAC (with sign extension)
- A source/destination for load/store operation

Bits 0-15 of a general register (L-suffix specified) are treated as 16-bit register when used as:

- Input operand to MAC (unsigned)
- Shift count for shift operation
- Source/destination for load/store operation
- Source/destination for a register transfer
- Destination for immediate operation
- Loop count for hardware loop instruction

8-Bit General Registers. Bits 32-39 of a general register (E-suffix specified) are treated as 8-bit registers when used as the source or destination for load/store operation.

Multiply-Accumulator (MAC)

The parallel architecture of the MAC unit allows three operand arithmetic operations. The MAC performs mixed arithmetic involving signed and unsigned fractional/integer data. In a single instruction cycle, the MAC unit can perform a 16-bit multiply and a 40-bit accumulation to produce a 40-bit result. The two 16-bit inputs to the multiplier and the 40-bit input to the accumulator must be held in general registers. A MAC input shifter (MSFT) can perform a 1-bit or 16-bit right shift on input data to the MAC.

Data ALU. The data ALU can perform addition, subtraction, comparison and logical operations on 40-bit input operands to produce 40-bit results. The input operands to the ALU must normally be general registers, but a source operand can be specified as an immediate constant in some cases. Arithmetic operations use fractional two's complement arithmetic.

Barrel Shifter (BSFT) and Shift-and-Count Circuit (SAC). The 40-bit barrel shifter operates on 40-bit two's complement data to produce a 40-bit result. The unit performs multiple-bit shifts in either direction in a single instruction cycle. The shift amount is specified either as an immediate constant or as the low 7-bits of a general register. The ALU supports floating-point normalization by means of an operation to compute the shift amount for normalizing data to 32-bit two's complement fractional format.

Round (RND) and CLIP operations are also supported. The RND operation rounds a 40-bit input to 16-bit two's complement format and stores the sign-extended result into a 40-bit general register. The CLIP operation forces the magnitude of a 40-bit fractional two's complement input to be no greater than 1.0 by truncation and stores the sign-extended result into a 40-bit general register.

Data Address Control Unit (DACU)

The DACU performs address storage and address calculations. See figure 2 for an illustration of the DACU. Data memory is divided into two sub-units, designated the X and Y data memory. The X and Y data memory areas can be accessed in parallel with each other and in parallel with the processor's other units.

Data Pointers (DP0 - DP7). The DACU has eight 16-bit data pointers, DP0 - DP7. DP0 - DP3 reference X data memory only. DP4 - DP7 reference Y data memory only. The data pointers contain addresses used as pointers to X and Y data internal/external memory. Each register may be read or written through the Transfer Bus.

Index Registers (DN0 - DN7). The DACU has eight 16-bit index registers, DN0 - DN7. These index registers can be used to adjust the data pointers following a memory access, with DNn modifing DPn. They may be read or written through the Transfer Bus.

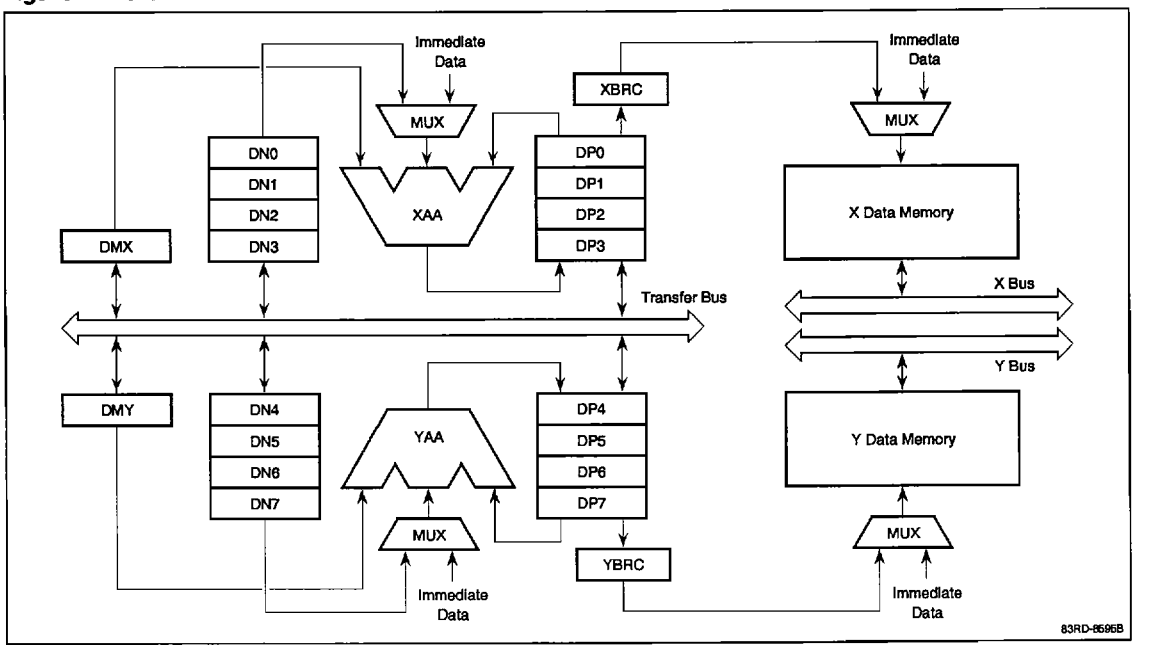
Modulo Registers (DMX, DMY). The X and Y data memory each have a 16-bit modulo register: DMX for X data memory and DMY for Y data memory. These registers specify modulo value for ring counting operations that use the data pointers and index registers. DMX and DMY may be read or written through the Transfer Bus.

Data Address Units (XAA, YAA). The two 16-bit data address units can operate in parallel to simultaneously modify two of the registers DP0 -DP7. XAA modifies the data pointers DP0 - DP3 and YAA modifies the data pointers DP4 - DP7.

Bit-Reverse Circuit (XBRC, YBRC). To support FFT calculations, the XBRC and the YBRC perform bit-reversing on addresses specified by DPn for X and Y data memory.

Data Addressing Modes. There are two categories of addressing modes, direct addressing and register indirect addressing. In direct addressing mode, the data address is contained within the instruction. In register indirect addressing mode, a data pointer, DPn, holds the data address.

Figure 2. Data Address Control Unit



In register indirect addressing mode, data pointers may be modified after a memory access in any of the following ways:

No Update	DPn is not modified after the current memory access.
Post-Increment	DPn is incremented by one after the current memory access.
Post-Decrement	DPn is decremented by one after the current memory access.
Indexed Addition	The content of DNn is added to DPn after the current memory access. Bit reversed memory access may be specified with this mode of post-access address modification.
Modulo Indexed Addition	The content of the DNn is added to DPn after the current memory access and the result is reduced modulo DMX or DMY as appropriate.
Immediate Addition	The immediate value specified within the instruction is added to DPn after the current memory access.

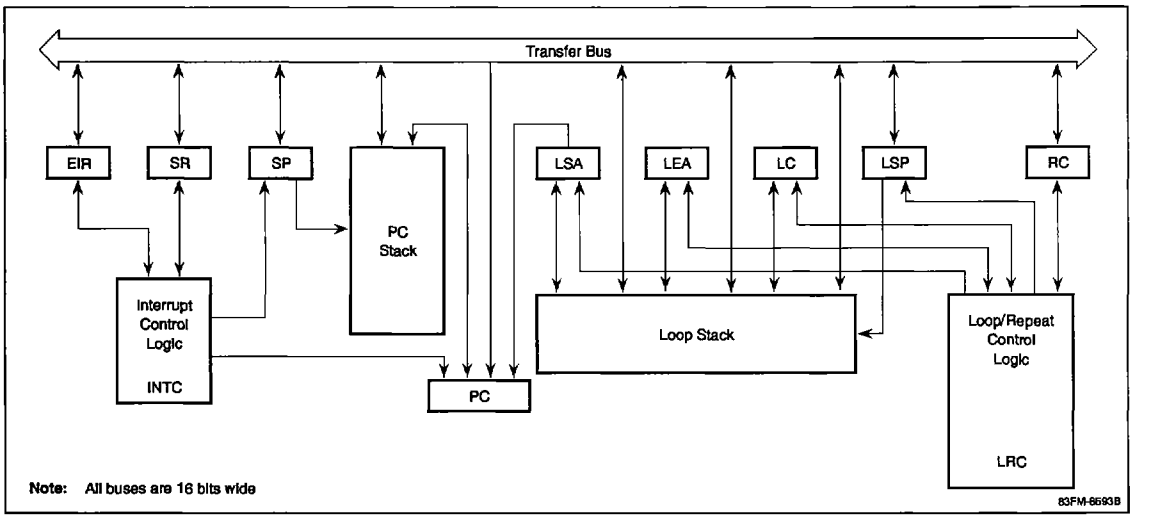
Program Control Unit

The Program Control Unit (PCU) performs instruction fetch, instruction decoding, HALT, branch control, exception processing, hardware no-overhead REPEAT and hardware no-overhead LOOP control. There is a four-level loop stack to support nested loops. This stack is separate from the 15-level PC stack that supports subroutine calls and interrupts. The PCU contains the following control registers:

PC	16-bit program counter
STK	15-level PC stack
SP	16-bit PC stack pointer
SR	Status register
EIR	Interrupt enable flag stack register
LSTK	4-level loop stack
LSP	16-bit loop stack pointer
LSA	16-bit loop start address register
LEA	16-bit loop end address register
LC	16-bit loop counter
RC	16-bit repeat counter

Figure 3 is a block diagram of the PCU.

Figure 3. Program Control Unit

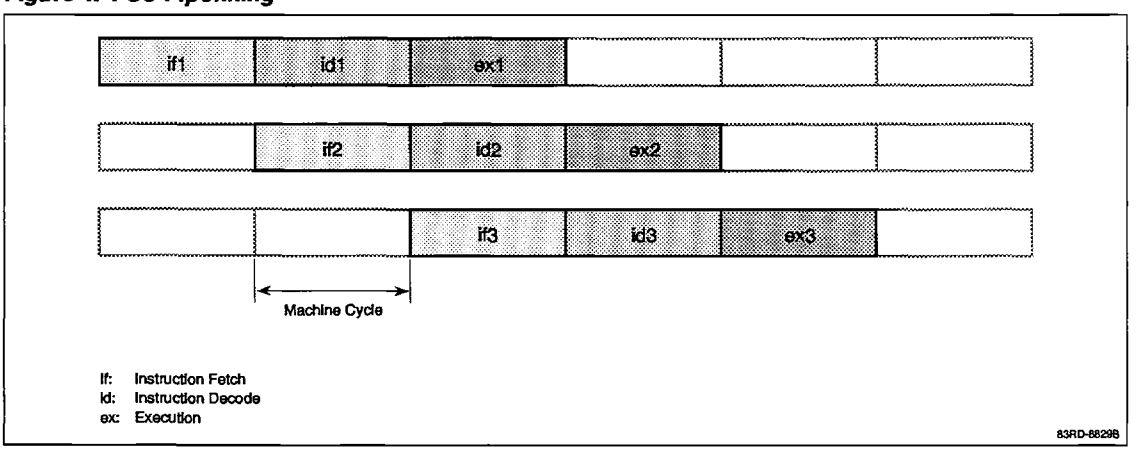


Pipelining. The basic operation of the PCU is performed in three pipeline stages that are functionally transparent to users. The three stages are:

if: Instruction Fetch
id: Instruction Decode
ex: Execution

Figure 4 illustrates pipelining.

Figure 4. PCU Pipelining



Program Counter (PC) and PC Stack (STK). The PC is a 16-bit counter that contains the address of the executing instruction. It is incremented upon each instruction fetch, except at branches and other program discontinuities. The STK is a 15-level, 16-bit register file that stores the PC for subroutine calls and interrupts.

In the case of jump operations, the new address is transferred to the PC. When a subroutine call or interrupt occurs, the contents of the PC are stored (pushed) onto the top location in the STK. When a return from a subroutine or interrupt occurs, the contents of the top location in the STK pop to the PC and the stack pointer is decremented by one.

Zero Overhead Loop and Repeat Control. The μPD77017 supports zero overhead loop and zero overhead repeat without requiring software to manage constructs such as delayed branches. The 16-bit loop counter (LC) specifies the number of times to iterate a program loop. The LC is set by LOOP instruction and may be read via the Transfer Bus.

The μPD77017 hardware directly supports nesting of loops up to a depth of four. Nested loops with a depth of five or more can be constructed in software. The LSTK is a four level 48-bit register file for storing the LSA, LEA, and LC. The LSP is a 16-bit register that points to the current level of the LSTK.

When a LOOP instruction starts to execute, the following occurs:

- LSP is incremented
- LSA, LEA and LC are pushed onto the LSTK
- The new LOOP start address is stored in the LSA
- The new LOOP end address is calculated and stored in the LEA
- LC is initialized with the value specified in the LOOP instruction

At the end of each loop, if LC ! = 1 then the following occurs:

- LC is decremented when the PC fetches the address specified in the LEA
- Value in the LSA is stored in the PC

If the LC is 1 at the end of a loop, the loop terminates and the following occurs:

- PC is incremented by one
- LSA, LEA, and LC whose location specified by the LSP are popped from the LSTK

For zero overhead repeat, there is a 16-bit repeat counter (RC) that specifies the number of times to repeat an instruction. The RC is set by the REPEAT instruction. Note that the REPEAT mechanism is separate from the LOOP mechanism to allow REPEATs to be nested within LOOPs. Repeats are not interruptable.

Interrupt Control Logic. The μPD77017 supports a separate interrupt vector for each of the 12 interrupt sources (4 external and 8 internal). Each interrupt source may be masked by setting bits of the Status Register (SR). The SR also stacks the interrupt history so that multiple interrupts can be handled.

After receiving an interrupt, the μPD77017 finishes executing the current instruction. Instruction decoding also completes before the interrupt is serviced. The instruction being fetched at the time of the interrupt normally will be aborted to be fetched after the interrupt is serviced. However, an additional instruction will be executed before the interrupt is serviced, if any of the following instructions are being decoded or executed:

- JUMP
- CALL
- RETURN
- RETURNI
- REPEAT
- LOOP
- Target instruction for REPEAT instruction
- Instruction specified in the LEA

In the case of REPEAT, all iterations are completed before the interrupt is serviced.

Memory Interface

The μPD77017 has a 64K x 32-bit instruction memory space and two separate 64K x 16-bit data memory spaces. There is a single external bus in the μPD77017 for external data memory. External instruction and data memory accesses may be independently extended by programmable wait-state control. Figure 5 illustrates the memory maps for both the instruction and data memory spaces.

Instruction Memory. An instruction RAM (IRAM) of 1.5K x 32 bits resides on the chip. Software for loading instruction code into the internal IRAM from external memory is held in a internal 256 x 32-bit boot ROM. The on-chip IRAM also has 64 locations allocated as interrupt vectors (some of these locations are reserved for use in future products).

Figure 5. Instruction Memory Map (Comparison of the μPD77016 and μPD77017)

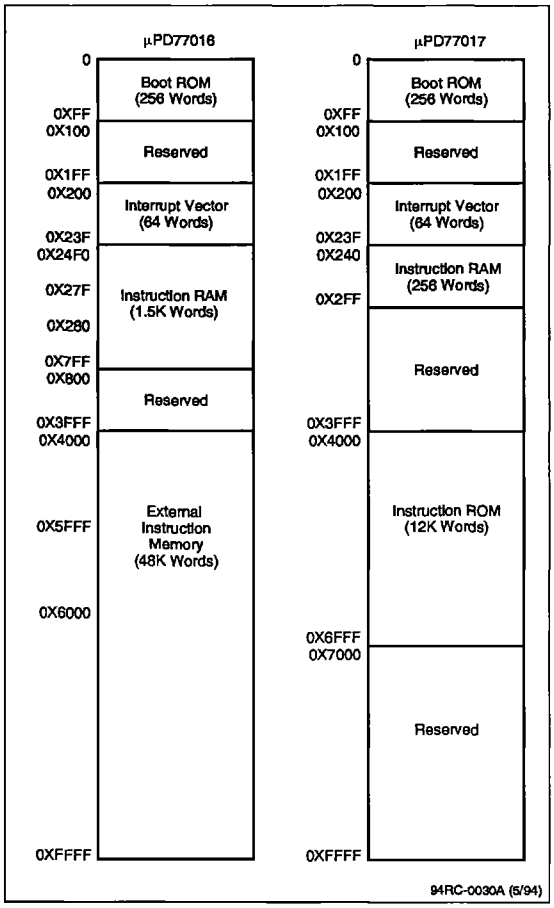
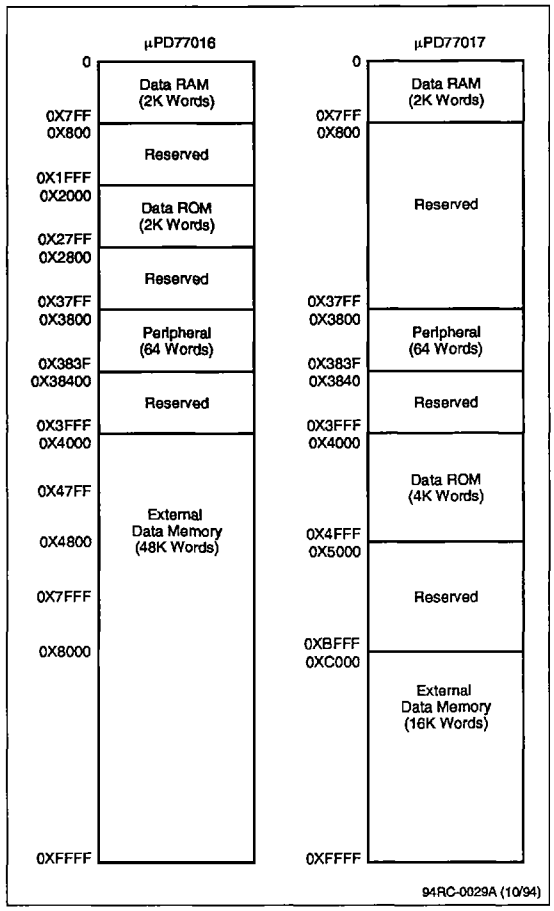


Figure 6. Data Memory Map (Comparison of the μPD77016 and μPD77017)



Data Memory. X and Y data memory are each 64K x 16-bit. A portion of each refers to on-chip 2K x 16-bit data RAM and 2K x 16-bit data ROM. Another 64 locations are assigned for controlling on-chip peripherals. Data from external memory may be accessed through a 16-bit external data bus either in a single cycle or with wait states that are configured by software. Avoid instructions that attempt two external data accessed because external Y memory will not be accessed during the same instruction cycle that X memory is accessed.

Bus Arbitration Support. The μPD77017 provides signal for bus arbitration to permit memory sharing by multiple μPD77017's, by a μPD77017 and a host CPU, or by a μPD77017 and a DMA.

Peripheral Unit

The following peripherals are incorporated in the μPD77017:

- Serial I/O
- Host CPU Interface
- General-Purpose I/O Port
- Wait-State Control
- Emulation-on-Chip

Registers in the peripheral units are mapped onto X and Y data memory and may be accessed as data.

Serial I/O. There are two identical full-duplex, double-buffered, serial I/O interfaces in the μPD77017. Each interface allows the device to communicate with any of a variety of codecs. Serial clock signals must be supplied externally for each serial channel. Transmit and receive frame lengths can be either 8-bit or 16-bit. Lengths are specified by software for the transmitter and receiver sides of each channel. Data can be received or transmitted in two ways: most significant bit (MSB) first or least significant bit (LSB) first. The μPD77017 can be programmed to service serial communications through interrupts, polling or simply waiting for the next received data word.

Host CPU Interface. The μPD77017 provides an 8-bit parallel port to communicate with either host CPU or a DMA controller. The host address input signal HA0 - 1 specifies access to the internal host data register and data for external communication. There are two 16-bit registers in the μPD77017 for transmitting and receiving data and status. These registers are mapped onto the data memory space. The μPD77017 can be programmed to service host communication through interrupts, polling or waiting for the next received data word.

General-Purpose I/O Ports. Four general purpose I/O pins can be programmed to serve as either input or output ports. A 16-bit port data register (twelve bits of which is un-used) receives input data from the input ports and sets data on the output ports. A 16-bit command register is used to program the ports. These registers are mapped onto the data memory space.

Wait-State Control. There are two 16-bit wait cycle registers that specify wait cycle counts for external-memory access (DWTR for data and IWTR for instruction). The data memory space is divided into eight areas and 0, 1, 3, or 7 wait states can be specified for each area. The instruction memory space is divided into four areas and 0, 1, 3, or 7 wait states can also be specified in each of these areas. External hardware can add wait states by asserting the $\overline{\text{WAIT}}$ signal.

Programming Model

Figure 7 illustrates the registers available for programming the μPD77017.

INSTRUCTION SET

The μPD77017 assembly language promotes efficient and optimized coding of DSP algorithms. The instruction set's key features include:

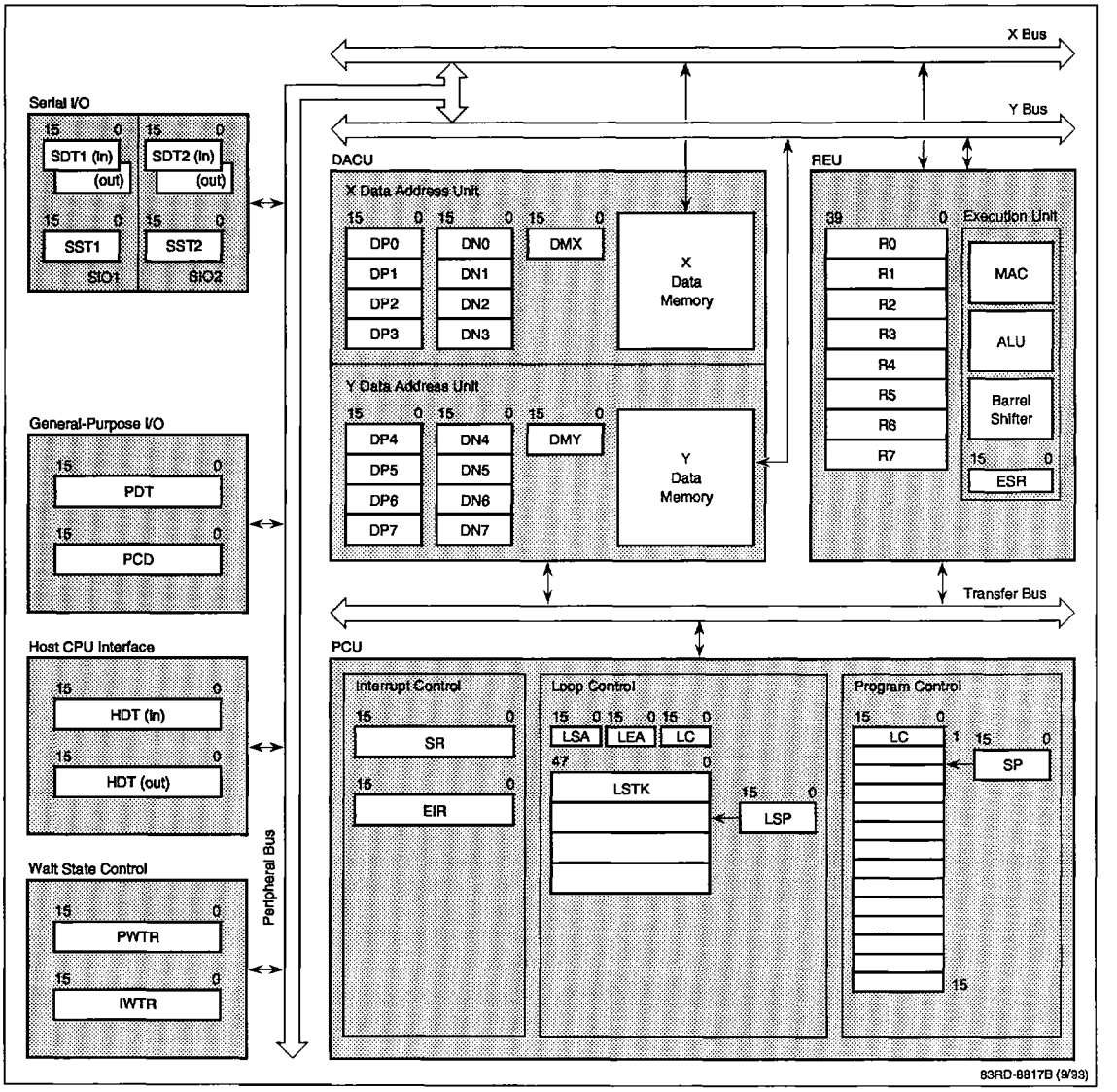
- C-like arithmetic assembler constructs
- Register-based arithmetic and logic operations
- Dual load/store operation concurrent with ALU and MAC operations
- Zero overhead looping and low overhead branching

Data Format

This section shows data formats for general registers (R0 - R7) as operands.

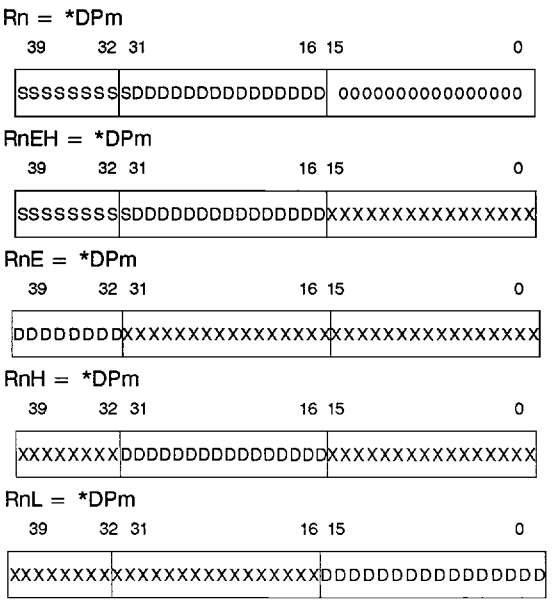
Fixed Point Fractional Data Format. This format offers 40-bit and 16-bit two's complement fractional data representations. In this case, the radix point is assumed to lie between bit 30 and bit 31.

Figure 7. Programming Model



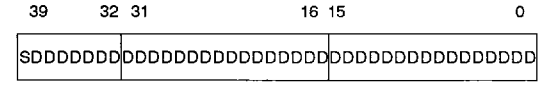
The 40-bit data format is illustrated in figures 8 and 9.

Figure 8. Loading from Data Memory



S: Sign bit
D: Data
X: Unchanged

Figure 9. Operation Result



S: Sign bit
D: Data
X: Unchanged

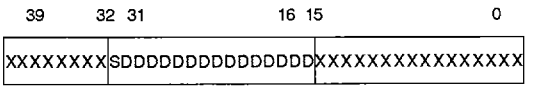
The results of the following operations are treated as a special case of the 40-bit data format.

- Unsign-unsign multiply add (UUMA)
- Logical operations
- Less than operations (LT)
- Exponent operations (EXP)

For discussions of these special cases, see the appropriate section in the μPD77017 User's Manual.

The 16-bit data format is shown in figure 10.

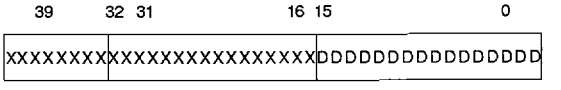
Figure 10. Multiplier Input



S: Sign bit
D: Data
X: Not used

Integer Format. The μPD77017 offers 16-bit data format for the general registers. The multiplier and shift amount input is shown in figure 11.

Figure 11. Multiplier and Shift Amount Input



S: Sign bit
D: Data
X: Not used

Instruction Format

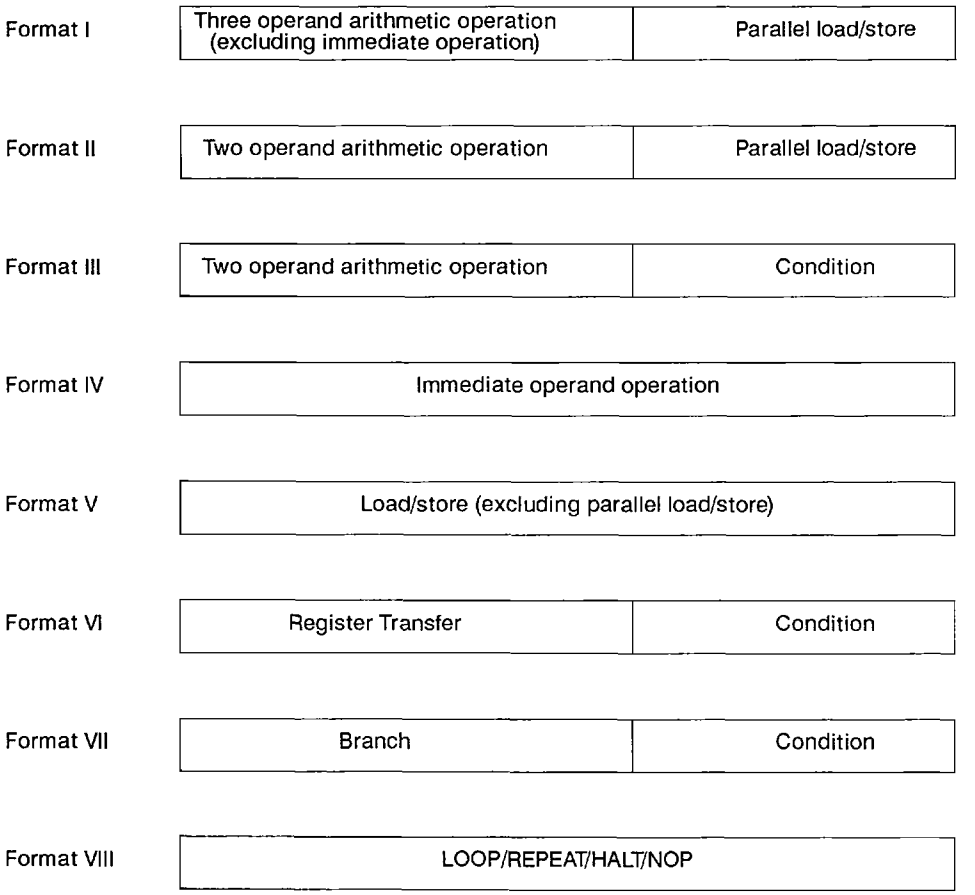
Instructions are 32 bits wide. Eight instruction formats are supported in the μPD77017. The assembly language uses a C-like arithmetic notation. Multiple operations can often be executed in a single instruction cycle. For example, the following instruction belongs to instruction format I:

R2 = R2 + R0H*R1H R0 = *DP0++ R1 = *DP4--;

In this case, a MAC is performed and three are two data transfers. Data specified by DP0 is transferred from X data memory into general register R0, and data specified by DP4 is transferred into R1. In addition, DP0 and DP4 are each incremented by one or decremented by one.

Figure 12 shows the list of instruction formats. These instruction formats are referenced in tables 4 to 12.

Figure 12. Instruction Formats



Instruction Categories

The μPD77017 provides nine operation categories that are described below. Most instructions are executed in a single cycle, however, loop and branch instructions may take two or three cycles.

Each operation category is outlined in a table. The notation conventions used in the operation category tables are described in tables 2 and 3. Many of the μPD77017 operations can be conditionally executed without additional execution time overhead. The possible conditions are all tests of individual general registers as detailed in table 13.

Three Operand Arithmetic Operations. Three operand arithmetic operations specify MAC operations. General registers must be specified for each of the three operands. Table 4 shows the three operand arithmetic operations.

Two Operand Arithmetic Operations. Two operand arithmetic operations specify MAC, ALU, or barrel shifter operations. Two general registers may be specified for operands. Table 5 shows the two operand arithmetic operations.

Immediate Operand Operations. Immediate operand operations specify ALU operations. One or two general register can be specified. Table 6 shows the immediate operand operations.

Load/Store Operations. Load/store operations specify data transfer between general register and data memory. Source and destination are specified by general registers on data pointers. Table 7 shows the load/store operations.

Register Transfer Operations. Register transfer operations specify data transfer between a general register and other registers. Table 8 shows the register transfer operations.

Immediate Load Operations. Immediate load operations store an immediate constant into a general register or store data from a register to a data address that is specified as immediate data. Table 9 shows the immediate load operations.

Branch Operations. Branch operations specify a program discontinuity. Branch location may be specified within the range of 32K location from the current value in the PC. Table 10 shows the branch operations.

Hardware Loop/Repeat Operations. The loop operation executes a block of operations repetitively. The repeat operation executes a single operation repetitively. Table 11 shows these operations.

Program Control Operations. Table 12 shows the program control operations.

Table 2. Notation and Selectable Register

Notation	Selectable Register
dm	DMX, DMY
dn	DN0 - DN7
dp	DP0 - DP7
dp_imm	DPn##imm (n = 0-7)
dpx	DP0 - DP3
dpx_imm	DPn##imm (n = 0-3)
dpx_mod	DPn, DPn+ +, DPn--, DPn##, DPn%%, !DPn## (n = 0-3)
dpy	DP4 - DP7
dpy_imm	DPn##imm (n = 4-7)
dpy_mod	DPn, DPn+ +, DPn--, DPn##, DPn%%, !DPn## (n = 4-7)
re	R0E - R7E
reh	R0EH - R7EH
rh, rh'	R0H - R7H
rl, rl'	R0L - R7L
ro, ro', ro"	R0 - R7
*XXXX:X (or Y)	Content of memory location addressed by XXXX

Table 3. Updating Data Pointer

Expression	Operation	Description
DPn	No change	
DPn+ +	DPn ← DPn+1	Post increment
DPn- -	DPn ← DPn-1	Post decrement
DPn##	DPn ← DPn+ DNn	Indexed addition
DPn%%	DPn ← residue of (DPn+ DNn)/DMX (n = 0-3)	Modulo indexed addition
	DPn ← residue of (DPn+ DNn)/DMX (n = 4-7)	
DPn##imm	DPn ← DPn+ imm	Immediate addition
!DPn##	First, data is accessed with bit-reverse DPn, and then DPn ← -DPn+ DNn	Bit reverse access

Table 4. Three Operand Arithmetic Operations (Note 1)

Name	Description	Assembler Mnemonic (Note 2)	Operation	Affects Overflow Flag	Format
ADD	Add	ro" = ro + ro'	ro" ← ro + ro'	Yes	I
AND	And	ro" = ro & ro'	ro" ← ro & ro'	No	I
LT	Less than	ro" = LT(ro, ro')	if (ro < ro') {ro' ← 0x0000000001} else {ro" ← 0x000000000}	No	I
MADD	Multiply add	ro = ro + rh * rh'	ro ← ro + rh * rh'	Yes	I
MAS1	1-bit shift multiply add	ro = ro >> 1 + rh * rh'	ro ← $ro/_2$ + rh * rh'	Yes	I
MAS16	16-bit shift multiply add	ro = ro >> 16 + rh * rh'	ro ← $ro/2^{16}$ + rh * rh'	No	I
MPY	Multiply	ro = rh * rh'	ro ← rh * rh'	No	I
MSUB	Multiply subtract	ro = ro – rh * rh'	ro ← ro – rh * rh'	Yes	I
OR	Or	ro" = ro | ro'	ro" ← ro | ro'	No	I
SLL	Logical left shift	ro' = ro SLL rl (Note 3)	ro' ← ro << r	No	I
SRA	Arithmetic right shift	ro' = ro SRA rl (Note 3)	ro' ← ro >> r	No	I
SRL	Logical right shift	ro' = ro SRL rl (Note 3)	ro' ← ro >> r	No	I
SUB	Sub	ro" = ro – ro'	ro" ← ro – ro'	Yes	I
SUMA	Signed unsigned multiply add	ro = ro + rh * rl (Note 3)	ro ← ro + rh * rl	Yes	I
UUMA	Unsigned unsigned multiply add	ro = ro + rl * rl' (Note 3)	ro ← ro + rl * rl'	Yes	I
XOR	Exclusive or	ro' = ro ^ ro'	ro" ← ro ^ ro'	No	I

Notes:

(1) Three operand arithmetic operations can be executed concurrently with XY parallel load/store operations.

(2) Two's complement arithmetic is used for all operations on two's complement data.

(3) The data in ro and rh are represented in two's complement notation, while rl and rl' are in positive integer notation.

Table 5. Two Operand Arithmetic Operations (Note 1)

Name	Description	Assembler Mnemonic (Note 2)	Operation	Affects Overflow Flag	Format
ABS	Absolute value	ro'= ABS(ro)	if (ro< 0) {ro'← −ro} else {ro'← ro}	Yes	II, III
ACA	Accumulate add	ro'+ = ro	ro'←ro' + ro	Yes	II, III
ACS	Accumulate subtract	ro'− = ro	ro'←ro' − ro	Yes	II, III
CLIP	Clip	ro'= CLIP(ro)	if (ro > 0x007FFFFFFF) {ro'← 0x007FFFFFFF} else if (ro < FF80000000) {ro'← 0xFF80000000} else {ro← (ro+ 0x8000) & 0xFFFFFF0000}	Yes	II, III
CLR	Clear	CLR(ro)	ro← 0H	No	II, III
DEC	Decrement	ro'= ro − 1	ro'← ro − 1	Yes	II, III

DIV	Divide (Note 3)	ro'/= ro	if (sign(ro')= = sign(ro))
			{ro'←(ro'-ro) < < 1}
			else {ro'←(ro'+ ro) < < 1}
			if (sign(ro')= = 0)
			{ro'←ro'= 1}
				Yes	II, III

EXP	Exponent	ro'= EXP(ro)	ro'= log2 $\left(\frac{1}{ro}\right)$	No	II, III
INC	Increment	ro'= ro+1	ro'= ro+1	Yes	II, III
NEG	Two's complement	ro'= −ro	ro'← −ro	Yes	II, III
NOT	One's complement	ro'= ~ro	ro' ← ~ro	No	II, III
PUT	Put	ro'= ro	ro'←ro	No	II, III
RND	Round	ro' = ROUND(ro)	if (ro > 0x007FFF0000) {ro ← 0x007FFF0000} else if (ro> 0xFF80000000) {ro' ← 0xFF80000000} else {ro' ← (ro + 0x8000) & 0xFFFFFF0000}	Yes	II, III

Notes:

(1) Two operand arithmetic operations can be executed conditionally. Alternatively, they can be executed concurrently with parallel load/store operations.

(2) The data in ro, rl and rh takes two's complement representation for arithmetic operations unless otherwise stated.

(3) The divide operation calculates only a single bit of the quotient and must be repeated sixteen times to calculate a 16-bit quotient.

Table 6. Immediate Operand Operations (Note 1)

Name	Description	Assembler Mnemonic (Note 2)	Operation	Affects Overflow Flag	Format
IADD	Immediate add	ro' = ro + imm	ro' ← ro + imm	Yes	IV
IAND	Immediate and	ro' = ro & imm	ro' = ro & imm	No	IV
IOR	Immediate or	ro" = ro I imm	ro" ← ro I imm	No	IV
ISLL	Immediate logical left shift	ro' = ro SLL imm (Note 3)	ro' ← ro < < imm	No	IV
ISRA	Immediate arithmetic right shift	ro' = ro SRA imm (Note 3)	ro' ← ro < < imm	No	IV
ISRL	Immediate logical right shift	ro' = ro SRL imm (Note 3)	ro' ← ro > > imm	No	IV
ISUB	Immediate sub	ro' = ro – imm	ro' ← ro – imm	Yes	IV
IXOR	Immediate exclusive or	ro' = ro ^ imm	ro' ← ro ^ imm	No	IV

Notes:

(1) Immediate operand operations cannot be executed conditionally and cannot be executed concurrently with any other operations.

(2) Data are represented in two's complement value.

(3) The data in ro and ro' are represented in two's complement notation, while imm is in positive integer notation.

Table 7. Load/Store Operations

Name	Description	Assembler Mnemonic (Note 1)	Operation	Affects Overflow Flag	Format
LSDA	Direct addressing load/store	dest = *addr	dest ← *addr	No	V
		*addr = source	*addr ← source		
LSIM	Immediate index load/store	dest = *dp_imm	dest ← *dp	No	V
		*dp_imm = source	*dp ← source		
LSSE	Section load/store (Note 2)	dest = *dpx_mod dest' = *dpy_mod	dest ← *dpx dest' ← *dpy	No	V
		dest = *dpx_mod *dpy_mod = source	dest ← *dpx *dpy ← source		
		*dpx_mod = source dest = *dpy_mod	*dpx ← source dest ← *dpy		
		*dpx_mod = source *dpy_mod = source'	*dpx ← source *dpy ← source'		
LSPA	Parallel load/store (Note 3)	ro = *dpx_mod ro' = *dpx_mod	ro ← *dpx ro' ← *dpy	No	I, II
		ro = *dpx_mod *dpy_mod = rh	ro ← *dpx *dpy ← rh		
		*dpx_mod = rh ro = *dpy_mod	ro' ← *dpx ro ← *dpy		
		*dpx_mod = rh *dpy_mod = rh'	*dpx ← rh *dpy ← rh'		

Notes:

(1) Dest and dest' data can be selected from the following: ro, reh, re, rh or rl. The source data can be selected from re, rh or rl, and addr can be selected from either 0: X-xFFFF for X memory or 0: Y-0xFFFF for Y memory. See table 2 for notation.

(2) Data pointers are modified after data memory access with the current data pointer value. Load/store is possible with the X memory only, the Y memory only, or both the X and Y memory.

(3) This operation can be executed concurrently with three operand and two operand operations.

Table 8. Register Transfer Operation (Note 1, 2)

Name	Description	Assembler Mnemonic	Operation	Affects Overflow Flag	Format
MOV	Register transfer	dest = rl	dest ← rl	No	VI
		rl = source	rl ← source		

Notes:

(1) Registers transfer operations can be conditional.

(2) The dest and source data can be selected from one of the following: dp, dn, dm, SR, EIR, STACK, SP, LC (source only), LSP, LSR1, LSR2, LSR3, and ESR.

Table 9. Immediate Load Operation

Name	Description	Assembler Mnemonic	Operation	Affects Overflow Flag	Format
LDI	Immediate value set	rl = imm (imm = 0 – 0xFFFF)	rl ← imm	No	V
		dp = imm (imm = 0 – 0xFFFF)	dp ← imm		
		dn = imm (imm = 0 – 0xFFFF)	dn ← imm		
		dm = imm (imm = 0 – 0x7FFF)	dm ← imm		

Table 10. Branch Operations (Note 1)

Name	Description	Assembler Mnemonic (Note 2)	Operation	Affects Overflow Flag	Format
CREG	Register indirect subroutine call	CALL dp	SP ← SP + 1 STK ← PC + 1 PC ← dp	No	VII
CREL	Relative subroutine call	CALL imm	SP ← SP + 1 STK ← PC + 1 PC ← PC + imm	No	VII
JREG	Register indirect jump	JUMP dp	PC ← dp	No	VII
JREL	Register jump	JUMP imm	PC ← PC + imm	No	VII
RET	Return	RETURN	PC ← STK SP ← SP – 1	No	VII
RETI	Return from interrupt	RETURNI	PC ← STK STK ← SP – 1 (Note 2)	No	VII

Notes:

(1) Branch operations can be conditonal.

(2) The interrupt mask flag is restored to its previous state.

Table 11. Hardware Loop/Repeat Operations

Name	Description	Assembler Mnemonic (Note 1)	Operation	Affects Overflow Flag	Format
LOOP	Loop	LOOP count	Start LSP ← LSP + 1 LSR1 ← LSA LSR2 ← LEA LSR3 ← LC LSA ← PC + 1 LEA ← PC + RLE (loop end relative address) LC ← count LF ← 0 Loop LC ← LC – 1 End PC ← LSA End PC ← PC + 1 LC ← LSR3 LEA ← LSR2 LSA ← LSR1 LSP ← LSP – 1	No	VIII
REP	Repeat	REPEAT count	Start RC ← count RF ← 0 During PC ← PC Repeat RC ← RC – 1 End PC ← PC + 1 RF ← 1	No	VIII

Notes:

(1) Count can be specified either as an immediate value or as the value of a general register. It can be any value from 1 to 0x7FFF.

Table 12. Program Control Operations

Name	Description	Assembler Mnemonic	Operation	Affects Overflow Flag	Format
COND	Condition	IF (ro cond)	Conditional judgment	No	II, VI, VII
HALT	Halt	HALT	CPU stop	No	VIII
LPOP	Loop pop	LPOP	LC ← LSR3 LEA ← LSR2 LSA ← LSR1 LSP ← LSP – 1	No	VIII
NOP	No operation	NOP	PC ← PC + 1	No	VIII
Stop	Stop	Stop	CPU Stop	No	VIII
FINT	Forget interrupts	FINT	Abandon pending interrupts	No	VIII

Table 13. General Register Ro Test Conditions

Condition	Description
EVER	Unconditional
ro= = 0	Test if content of ro equal to zero
ro!= 0	Test if content of ro is not equal to zero
ro> 0	Test if content of ro is greater than zero
ro< 0	Test if content of ro is less than zero
ro< = 0	Test if content of ro is less than or equal to zero
ro> = 0	Test if content of ro is greater than or equal to zero
ro= = EX	Test if bit pattern 31-39 of ro is mixture of 1's and 0's
ro!= EX	Test if bit pattern 31-39 of ro all 1's or all 0's

DEVELOPMENT TOOLS

The μPD77016 is supported by hardware emulation tools and DOS and Windows-based software development tools.

IE-77016-PC-EM1 — An ISA bus board that plugs into a PC to support hardware debugging. It is actually a communications board that allows the PC to talk to the debugging unit on the μPD77016 chip on a target system. The effect is to provide in-circuit emulation capability from the PC.

IE-77016-CM-EM6 — A daughter card for the IE-77016-PC-EM1 that serves as an optional target system. In this way, the two boards installed in a PC can both execute and debug μPD77016 code with no need for an external target system.

IE-77017 — The IE-77017 is a system that includes a μPD77016, voltage level converters, high-speed RAM and a 66 MHz clock. The primary purpose of this system is to emulate a μPD77017 at full speed.

Alternatively, the IE-77017 can be used to interface the μPD77016/EM1 to a μPD77017 target system (which has a μPD77017 installed). In this way, the IE-77016 can be used to debug μPD77017 hardware.

WB77016 — An integrated development environment that includes a relocatable macro assembler, a linker, a librarian, and an editor. Both the simulator and the emulator can be invoked from within the WB77016 workbench.

CC77016 — Windows-based optimizing C compiler that generates assembly code output. The compiler is ANSI C-compliant and includes extensions to support DSP-specific requirements. A Windows interface for the compiler is provided

SM77016 — A full-featured, user-friendly Windows-based simulator.

Hardware Tools

Since the μPD77017 is a mask ROM part, early prototyping is generally done using a μPD77016, possibly within an IE-77017.

Each μPD7701x has built-in emulation support to provide real-time emulation on a developer's target board. Developers can debug a DSP application at 66 MHz clock frequency with only the IE-77016 PC plug-in board and a IE-77017 mounted on the developer's target board. The IE-77016 eliminates propagation delays and noise problems associated with remote emulators and probes.

The IE-77016 is easy to use. The IE-77016 board is simply plugged into the developer's PC. A six-wire interface cable connects to the developer's target board possibly through an IE-77017 via a small socket on that board. The interface uses IEEE's IE-77017 JTAG communications protocol. The cable itself is a full meter long so that a developer's PC need not be inconveniently close to his or her target board.

The IE-77016 offers normal monitor features plus a special debugging feature, the no-break monitor. The no-break monitor lets developers read or write a register or memory location without suspending execution.

Despite its small size, the IE-77016 has powerful debugging functions. IE-77016 can break program execution at an indicated program address or during access to a data memory address. A repetition count can modify any break condition. Developers can elect to break when an error flag is set in the error register. The IE-77016 also supports step execution.

Although developers require only the IE-77016 main board to debug a μPD77017 system, the IE-77016 main board can be supplemented with optional NEC-supported boards.

The IE-77017 is a low-cost alternative to expensive standalone emulators. The IE-77017 also has the potential to debug multiple CPUs.

Event Conditions

- Instruction address: 1 address location.
- X memory address: 1 location for read, write, or read/write; conditions selectable.
- Y memory address: 1 location for read, write, or read/write; conditions selectable.

Above three event conditions can be used with iteration counter (range; 0 x 1 - 0 x FFFF).

- When one of error register bits is set.
- Force event.

IE-77016 Features

- Clock frequency: 33 MHz max
- Break and step operation in real-time emulation using above conditions
- Debugged code can reside in ROM. There is no need to modify code while debugging
- Registers can be read without suspending execution
- Synchronous start allows an emulation to be started synchronously with user's system reset signal
- Power-on reset
- Emulation memory:
  — IE-77016-PC: 0 words
  — IE-77016-CM-EM4 (optional daughter board for the IE-77016-PC). Instruction RAM; 8K words, data RAM: 4K x 2 words
- Interface cable length: 1 meter
- Interface cable connector size: 2.54 x 5 x 2 mm

Software Tools

The μPD77017 is supported by a complete package of Windows-based software tools. The software tools include a simulator (SM77016) and workbench (WB77016). Within the workbench, a relocatable assembler, linker, and librarian can be accessed within the graphical interface. Like all Windows programs, repetitive actions (such as mouse clicks and key strokes) can be stored as macros.

A typical edit and debug session includes invoking the assembler, linker, and simulator. The simulator lets developers quickly and conveniently create DSP algorithms. The simulator inputs load modules from the linker's output (.LNK files). The simulator also lets developers arrange the windows in any convenient screen display. A typical screen display shows the following items: all eight working registers, instruction memory, X and Y RAM memory, and status. Simulator setups can be saved as a SIM file for quick reentry. See figure 12 for a sample screen display.

The C-like operations are typically shown in their disassembled form. The Fetch (F), Decode (D) and Execute (E) instructions are highlighted in the Instruction Memory window. The X and Y RAM location pointers are also highlighted. If a location has been read or written, a highlighted "R" for read or "W" for written will appear. The status register counts and displays instruction cycles, steps and execution time.

Run options allow a single step forward or backward and the simulator can be set to run with or without register display updates (animation).

Developers can set or clear breakpoints with a mouse click. Of the various breakpoint options, one can be set to occur after a specified execution time has elapsed.

The contents of the eight working registers can be formatted in various ways. Figure 12 shows the 40 bits formatted as 8 extension bits, 16 high data bits and 16 low data bits. Register values can also be evaluated as fractional numbers. Developers can select many other registers for display. Some examples include:

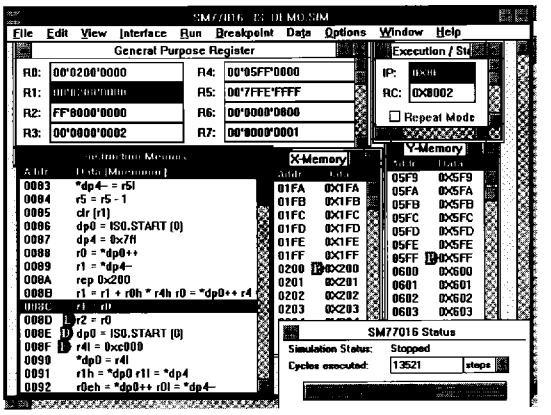
Status registers
Stack
Stack pointer
Loop start and end
Loop count
Repeat count
Error
Interrupt enable
Serial ports 1 and 2
X and Y memory pointers
X and Y modulo counter
Status of I/O pins

Developers can use timing files to simulate data input, output, and other data such as register values relating to execution. Multiple timing files can be concurrently active. Each timing file can input and output data to and from any number of data files. Nearly every simulation event can be used for synchronization. Timing files employ a highly structured language containing IF-THEN-ELSE, REPT, and DO blocks.

Simulator Features

- Simulation of all μPD77017 instructions
- Simulation of all I/O functions
- Complete Windows environment for relocatable assembler, linker, librarian, and simulator
- Complete register display options with selectable formats
- Simultaneous display of instruction memory and both internal data memories; all memory can be edited
- Instruction memory displayed as source code or hex
- Convenient point-and-click program counter breakpoint setting/clearing
- Breakpoint setting options:
  - Program counter
  - Elapsed time or number of steps
  - Memory access to a specific location or range of locations
  - When global expression evaluated to TRUE
- Execution options:
  - Run
  - Clock trace
  - Trace
  - Step
  - Animate on clock trace, trace, or step
  - Run until key input
  - Run to cursor
  - Backstep
- Cycle counting and timing
- Multiple timing and data files

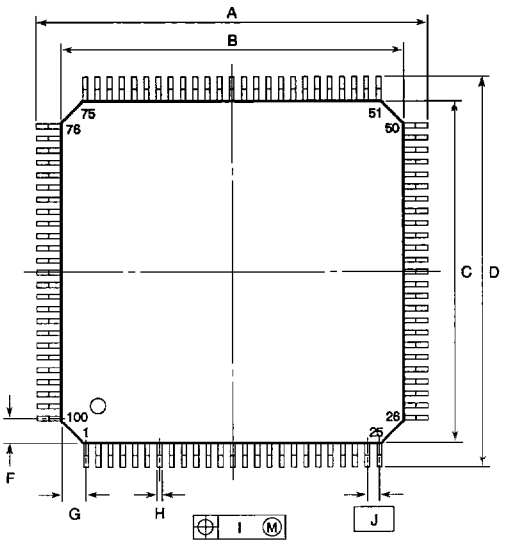
Figure 12. Sample Simulator Screen

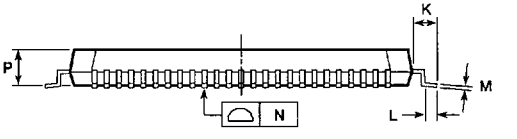


Package Drawing

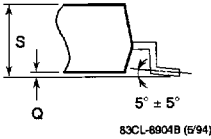
100-Pin Plastic Thin QFP (TQFP)

Item	Millimeters	Inches
A	16.0 ± 0.4	0.630 ± 0.016
B	14.0 ± 0.2	0.551 $^{+0.009}_{-0.008}$
C	14.0 ± 0.2	0.551 $^{+0.009}_{-0.008}$
D	16.0 ± 0.4	0.630 ± 0.016
F	1.0	0.039
G	1.0	0.039
H	0.20 ± 0.10	0.008 ± 0.004
I	0.08	0.003
J	0.5 (T.P.)	0.020 (T.P.)
K	1.0 ± 0.2	0.039 $^{+0.009}_{-0.008}$
L	0.5 ± 0.2	0.020 $^{+0.008}_{-0.009}$
M	0.125 ± 0.05	0.005 ± 0.002
N	0.10	0.004
P	1.0	0.039
Q	0.1 ±0.1	0.004 ± 0.004
S	1.27 max	0.049 max





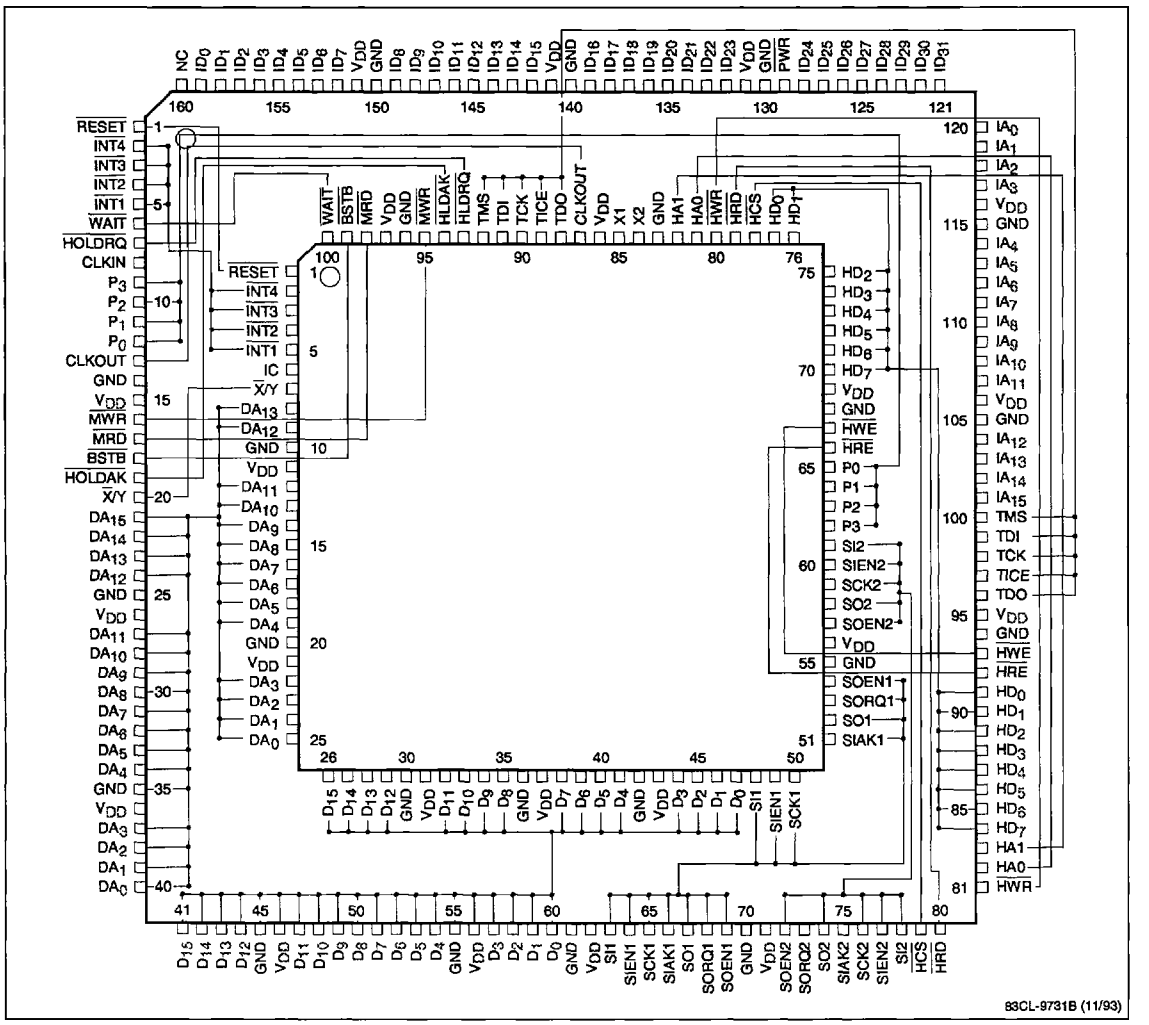
Enlarged detail of lead end



S100GC-60-9EU

83CL-8904B (5/94)

Pin/Signal Relationship of μPD77016 to μPD77017



μPD77017/77016 Pinout Correlation

μPD77017		μPD77016	
Pin No.	Description	Pin No.	Description
1	$\overline{\text{RESET}}$	1	$\overline{\text{RESET}}$
2	$\overline{\text{INT4}}$	2	$\overline{\text{INT4}}$
3	$\overline{\text{INT3}}$	3	$\overline{\text{INT3}}$
4	$\overline{\text{INT2}}$	4	$\overline{\text{INT2}}$
5	$\overline{\text{INT1}}$	5	$\overline{\text{INT1}}$
6	IC	N/A	N/A
7	$\overline{\text{X/Y}}$	20	$\overline{\text{X/Y}}$
8	DA13	23	DA13
9	DA12	24	DA12
10	GND		
11	V_{DD}		
12	DA11	27	DA11
13	DA10	28	DA10
14	DA9	29	DA9
15	DA8	30	DA8
16	DA7	31	DA7
17	DA6	32	DA6
18	DA5	33	DA5
19	DA4	34	DA4
20	GND		
21	V_{DD}		
22	DA3	37	DA3
23	DA2	38	DA2
24	DA1	39	DA1
25	DA0	40	DA0
26	D15	41	D15
27	D14	42	D14
28	D13	43	D13
29	D12	44	D12
30	GND		
31	V_{DD}		
32	D11	47	D11
33	D10	48	D10
34	D9	49	D9
35	D8	50	D8
36	GND		
37	V_{DD}		
38	D7	51	D7
39	D6	52	D6
40	D5	53	D5

μPD77017/77016 Pinout Correlation (cont)

μPD77017		μPD77016	
Pin No.	Description	Pin No.	Description
41	D4	54	D4
42	GND		
43	V_{DD}		
44	D3	57	D3
45	D2	58	D2
46	D1	59	D1
47	D0	60	D0
48	SI1	61	SI1
49	SIEN1	64	SIEN1
50	SCK1	65	SCK1
51	SIAK1	66	SIAK1
52	SO1	67	SO1
53	SORQ1	68	SORQ1
54	SOEN1	9	SOEN1
55	GND		
56	V_{DD}		
57	SOEN2	72	SOEN2
58	SO2	74	SO2
59	SCK2	76	SCK2
60	SIEN2	77	SIEN2
61	SI2	78	SI2
62	P3	9	P3
63	P2	10	P2
64	P1	11	P1
65	P0	12	P0
66	$\overline{\text{HRE}}$	92	$\overline{\text{HRE}}$
67	$\overline{\text{HWE}}$	93	$\overline{\text{HWE}}$
68	GND		
69	V_{DD}		
70	HD7	84	HD7
71	HD6	85	HD6
72	HD5	86	HD5
73	HD4	87	HD4
74	HD3	88	HD3
75	HD2	89	HD2
76	HD1	90	HD1
77	HD0	91	HD0
78	$\overline{\text{HCS}}$	79	$\overline{\text{HCS}}$
79	$\overline{\text{HRD}}$	80	$\overline{\text{HRD}}$
80	$\overline{\text{HWR}}$	81	$\overline{\text{HWR}}$

μPD77017/77016 Pinout Correlation

μPD77017		μPD77016	
Pin No.	Description	Pin No.	Description
81	HA0	82	HA0
82	HA1	83	HA1
83	GND		
84	X2		
85	X1	8	CLKIN
86	V_{DD}		
87	CLKOUT	13	CLKOUT
88	TDO	96	TDO
89	TICE	97	TICE
90	TCK	98	TCK
91	TDI	99	TDI
92	TMS	100	TMS
93	$\overline{HLDRQ}$	7	$\overline{HLDRQ}$
94	$\overline{HLDAK}$	19	$\overline{HLDAK}$
95	$\overline{MWR}$	16	$\overline{MWR}$
96	GND		
97	V_{DD}		
98	$\overline{MRD}$	17	$\overline{MRD}$
99	$\overline{BSTB}$	18	$\overline{BSTB}$
100	$\overline{WAIT}$	6	$\overline{WAIT}$
N/A	N/A	14	GND
N/A	N/A	15	V_{DD}
N/A	N/A	21	DA15
N/A	N/A	22	DA14
N/A	N/A	25	GND
N/A	N/A	26	V_{DD}
N/A	N/A	35	GND
N/A	N/A	36	V_{DD}
N/A	N/A	45	GND
N/A	N/A	46	V_{DD}
N/A	N/A	55	GND
N/A	N/A	56	V_{DD}
N/A	N/A	61	GND
N/A	N/A	62	V_{DD}
N/A	N/A	70	GND
N/A	N/A	71	V_{DD}
N/A	N/A	73	SORQ2
N/A	N/A	75	SIAK2
N/A	N/A	94	GND
N/A	N/A	95	V_{DD}

μPD77017/77016 Pinout Correlation (cont)

μPD77017		μPD77016	
Pin No.	Description	Pin No.	Description
N/A	N/A	96	NC
N/A	N/A	101	IA15
N/A	N/A	102	IA14
N/A	N/A	103	IA13
N/A	N/A	104	IA12
N/A	N/A	105	GND
N/A	N/A	106	V_{DD}
N/A	N/A	107	IA11
N/A	N/A	108	IA10
N/A	N/A	109	IA9
N/A	N/A	110	IA8
N/A	N/A	111	IA7
N/A	N/A	112	IA6
N/A	N/A	113	IA5
N/A	N/A	114	IA4
N/A	N/A	115	GND
N/A	N/A	116	V_{DD}
N/A	N/A	117	IA3
N/A	N/A	118	IA2
N/A	N/A	119	IA1
N/A	N/A	120	IA0
N/A	N/A	121	ID31
N/A	N/A	122	ID30
N/A	N/A	123	ID29
N/A	N/A	124	ID28
N/A	N/A	125	ID27
N/A	N/A	126	ID26
N/A	N/A	127	ID25
N/A	N/A	128	ID24
N/A	N/A	129	$\overline{PWR}$
N/A	N/A	130	GND
N/A	N/A	131	V_{DD}
N/A	N/A	132	ID23
N/A	N/A	133	ID22
N/A	N/A	134	ID21
N/A	N/A	135	ID20
N/A	N/A	136	ID19
N/A	N/A	137	ID18
N/A	N/A	138	ID17
N/A	N/A	139	ID16

μPD77017/77016 Pinout Correlation (cont)

μPD77017		μPD77016	
Pin No.	**Description**	**Pin No.**	**Description**
N/A	N/A	140	GND
N/A	N/A	141	V_{DD}
N/A	N/A	142	ID15
N/A	N/A	143	ID14
N/A	N/A	144	ID13
N/A	N/A	145	ID12
N/A	N/A	146	ID11
N/A	N/A	147	ID10
N/A	N/A	148	ID9
N/A	N/A	149	ID8
N/A	N/A	150	GND
N/A	N/A	151	V_{DD}
N/A	N/A	152	ID7
N/A	N/A	153	ID6
N/A	N/A	154	ID5
N/A	N/A	155	ID4
N/A	N/A	156	ID3
N/A	N/A	157	ID2
N/A	N/A	158	ID1
N/A	N/A	159	ID0

NEC

NEC Electronics Inc.

CORPORATE HEADQUARTERS

475 Ellis Street
P.O. Box 7241
Mountain View, CA 94039
TEL 415-960-6000

For literature, call toll-free 7 a.m. to 6 p.m. Pacific time:
1-800-366-9782
or FAX your request to: **1-800-729-9288**