ENSONIQ™

ES 5510 "ESP"

ENSONIQ Signal Processor

ENSONIQ ESP Specification Rev. 2.4

Table of Contents

List of Figures

ESP ES5510

1. Introduction

The ENSONIQ ESP 5510 is a VLSI device designed in a 1.0 micron double-metal CMOS process. It is optimized for signals in the audio frequency range, which have typical sample rates of between 10 kHz and 50 kHz. The ESP executes its microprogram once every sample period. It's nominal instruction cycle is 250ns, yielding program lengths from about 64 to 160 microinstructions at typical sample rates.

The major features of the ESP chip are:

- 48 Pin DIP or 52 Pin PLCC
- Separate Address Generator ALU
- 4 Programmable Serial I/O Channels (I2S or Sony Format)
- On Chip Data and Microprogram Memory
- 8 Bit Address/Data Multiplexed Host CPU Interface
- External Sample Rate Synchronization
- Multiplexed Addressing for Simple DRAM Interface
- Host Access to ESP DRAM

2. General Description

The architecture of the ESP chip is implemented by the following major components:

ALU
Multiplier
Address Generator
On-chip data and microprogram memory
Three 24 bit wide data paths
Stereo serial digital audio interface
Host interface

The ALU is a general purpose 24-bit wide arithmetic/logic unit capable of 16 different instructions. It has an associated Condition Code register which can be used to implement flow control constructs via the conditional execution of instructions.

The Multiplier is a 24 X 24 bit multiplier with its own dedicated 48 bit accumulator. It can produce an accumulated product every microinstruction cycle.

The Address Generator is a separate ALU dedicated to calculating RAM memory addresses. A new read or write address is calculated according to one of several addressing modes every microinstruction cycle. The Address Generator allows memory to simultaneously support both recirculating delay lines and fixed data table structures. Through the Address Generator, the chip will interface directly to up to 16 Kbytes of external 16-bit wide RAM. Data transfers to memory are performed through a Data Input Latch (DIL) and Data Output FIFO (DOL FIFO). The DOL FIFO is 2 words deep, to

improve output throughput. An alternate I/O memory space can also be selected on a cycle by cycle basis which can be used as a 16-bit parallel interface to external hardware.

On chip memory consists of 192 24-bit wide GPRs, or general purpose registers, and 160 45-bit microinstructions, implemented using internal dynamic RAM. In addition to the GPRs there are a number of SPRs, or special purpose registers, which are used for I/O, condition codes, and addressing control. The GPRs are used for the storage of RAM address offsets, multiplier coefficients, and as general scratch pad memory.

Three internal 24 bit data busses, X, Y, and Z connect the GPRs to the ESP's processing units. The X and Y busses provide the inputs to the ALU and Multiplier while the Z bus is used to write the results from the ALU and Multiplier back into the registers or to external memory.

Pins are provided to synchronize program execution and communication, allowing systems to be easily configured with a variety of other signal processors or sources. The synchronization mechanism also allows execution in single sample steps to facilitate the debugging of microprograms. Registers are provided on the ESP to interface the chip to a Host microprocessor required to load microprograms and data into internal memories. Typically the Host will also alter the contents of GPRs or microinstructions while microprograms are executing.

The ALU, Multiplier and Address Generator all execute concurrently, using the data busses at different times. A separate microprogram bus is used to supply microinstructions to the ESP's processing elements so that instructions and operands can be fetched simultaneously. The microprogram memory does not appear on the X, Y or Z busses, so it may not be modified directly by ESP microprograms.

The GPRs are the operands most often used for ALU and Multiplier operations. The contents of 4 distinct 24-bit GPRs can be fetched directly during each microinstruction cycle with no setup overhead. The results are written back to 1 or 2 of these 4 GPR source locations. Alternatively, external RAM accesses can be substituted for 1 or 2 of the source operands and for 1 of the destination operands. The multiplicity and flexibility of the data paths in the ESP allows many DSP operations to be accomplished in a minimum number of microinstruction steps. Only a moderate amount of pipelining exists in the ESP's internal data paths, enhancing its ease of programmability.

Examples	
Nth-order FIR filter	N steps
2nd Order Filter section	5 steps
All-pass filter	2 steps

3. Chip Architecture

The following are the major elements of the ESP chip:

Microinstruction Memory Array (160 x 45 bits)
General Purpose Register Array (192 x 24 bits)
X,Y,Z Bus System
Arithmetic/Logic Unit
Multiplier

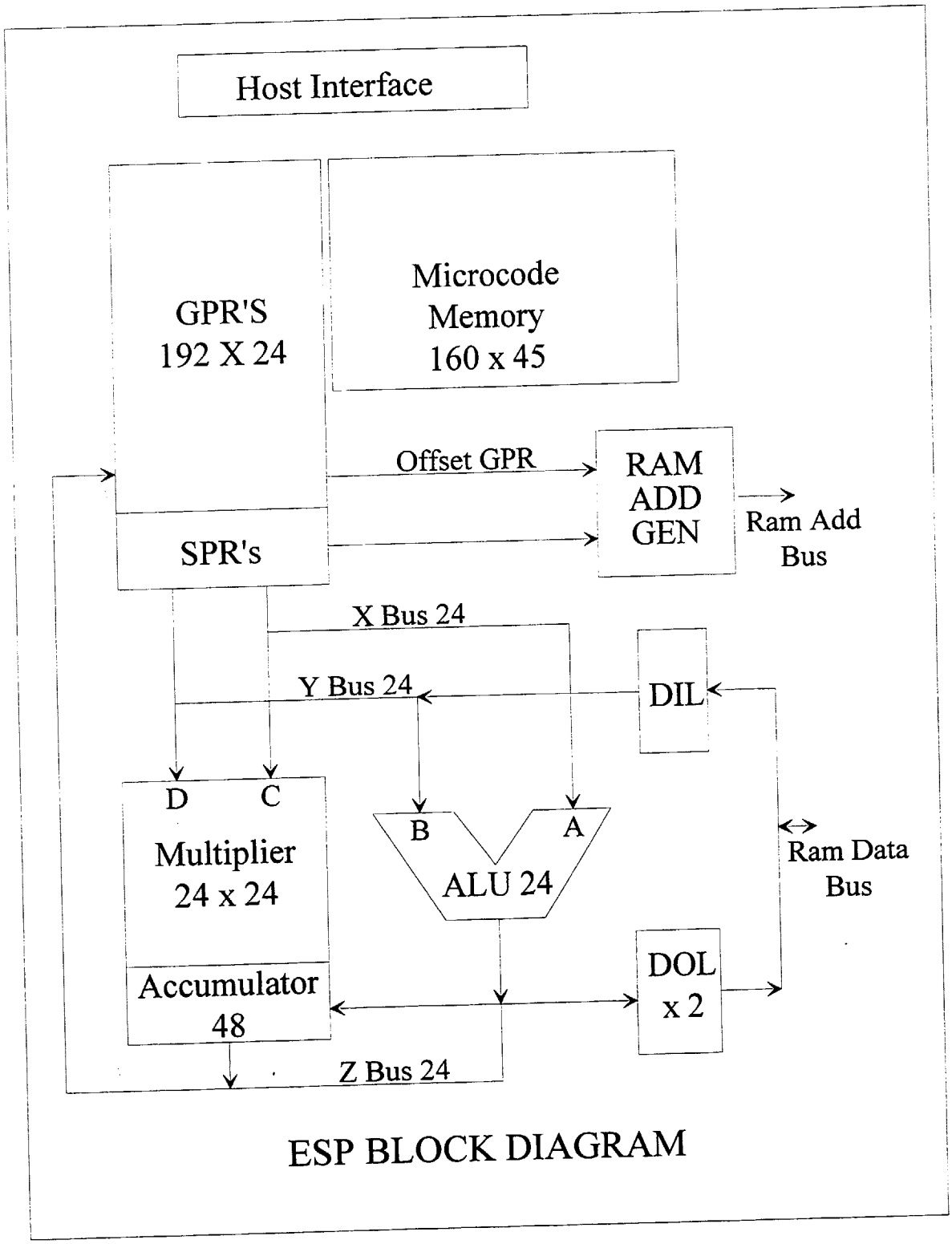
External Memory Address Generator
Host Interface
Special Purpose Registers
Serial Digital I/O (4 stereo channels, I2S or Sony)

3.1. Instruction and GPR Memory Arrays

The main memory inside the ESP is composed of 192 GPRs (general purpose registers) and 160 microinstruction words. These two memory arrays are arranged side by side, both physically and logically, so that each of the first 160 GPRs have an adjoining microinstruction word. The importance of this alignment will be discussed in the description of the Address Generator. Each GPR is 24 bits wide, while each Instruction word is 45 bits. Data in GPRs can be read and written by programs executing on the ESP, but the instruction memory can be accessed only by the ESP's program counter and by the Host CPU.

Instruction Memory is similar to any other microprocessor instruction storage area. It contains the program that will be executed by the ESP chip. The program will begin executing the first instruction in memory at the beginning of each sample period. The program will continue to execute each instruction in order until an END instruction is encountered at which point it will return to the first instruction and resume execution or wait for an external synchronization pulse to begin execution. GPRs have two purposes in the ESP chip, external RAM address offsets and data storage.

When used as an offset, the data in the GPR corresponding to the microinstruction is used to calculate the indirect addresses to external DRAM. GRPs not used to store offsets are available to hold multiplier coefficients or temporary variables used in the ESP program.



ESP BLOCK DIAGRAM

3.2. Special Purpose Registers

In addition to the GPR's, there are other 'hardware' registers which are addressed in the same way but have special functions. Among these special registers are the two 24 bit halves of

the multiplier's 48 bit product accumulator. Others include address generator table registers, and various machine control registers.

Special Purpose Registers

Address (dec)	Address (hex)	Width (bits)	Register Name	Function
Serial I/O Registers				
234	EA	16	SER0R	Right Serial I/O Register 0
235	EB	16	SER0L	Left Serial I/O Register 0
236	EC	16	SER1R	Right Serial I/O Register 1
237	ED	16	SER1L	Left Serial I/O Register 1
238	EE	16	SER2R	Right Serial I/O Register 2
239	EF	16	SER2L	Left Serial I/O Register 2
240	F0	16	SER3R	Right Serial I/O Register 3
241	F1	16	SER3L	Left Serial I/O Register 3
Multiplier Registers				
242	F2	24	MACL	Least Significant Multiplier Accumulator
243	F3	24	MACH	Most Significant Multiplier Accumulator
Address Generator Registers				
244	F4 (R)	16	DIL	Ram Data Input Latch (left justified)
244	F4 (W)	20	MEMSIZ	System Memory Size Register
245	F5	20	DLENGTH	Length of Delay Memory
246	F6	20	ABASE	Table "A" Base Address Register
247	F7	20	BBASE	Table "B" Base Address Register
248	F8	20	DBASE	Delay Base Address Counter (decrements from DLENGTH to 0)
Control Registers				
249	F9	1	SIGREG	ESP Signal Register Bit 22=Product Shift 1⇒Shift ONE, 0⇒ Shift TWO
250	FA	5	CCR	Condition Code Register
251	FB	6	CMR	Condition Mask Register
Constant Memory				
252	FC	24	$FFFFFF	-1 Constant (MINUS 1)
253	FD	24	$800000	Negative Full Scale (MIN)
254	FE	24	$7FFFFF	Positive Full Scale (MAX)
255	FF	24	$000000	Zero (ZERO)

The MACH and MACL registers are the most significant and least significant halves of the Multiplier's 48-bit wide accumulator. They can be accessed exactly as GPRs with few restrictions. See section 4.3.3.

The DIL register is the external RAM Data Input Latch, and is a special purpose register to enable access during 1-operand instructions. Only the leftmost 16 bits are used.

The unused 8 bits are read back as zeros. The DOL FIFO, mentioned previously, is not available to the host CPU. It is only available as a destination in an ESP instruction.

The CCR and CMR registers are concerned with conditional execution and are explained in Sections 3.2.2 and 4.2.The DLENGTH, ABASE, BBASE, DBASE, and MEMSIZ registers are all part of the Address Generator and are covered in Section 3.5. Writing the MEMSIZ register affects the DBASE register directly.

The arithmetic constants are often used by ESP programs. They are implemented in ROM, but can be written to when dummy destinations are needed.

3.3. The ALU

The Arithmetic/Logic Unit (ALU) can perform a variety of general and special purpose arithmetic, data movement, and logical operations. It incorporates optional zero-overhead saturation arithmetic for handling computational overflow (analogous to "clipping"), and can shift signals left or right for the purposes of normalization or amplification.

During every microinstruction cycle the ALU takes one or two 24-bit operands and produces a 24 bit result. ALU execution overlaps with the operation of the Multiplier, so that the two units may operate in parallel. Detailed description of the ALU and Multiplier execution cycles may be found in Section 4.3.

Associated with the ALU is a Condition Code Register (CCR) whose status bits are set as a result of ALU operations. The CCR along with the Condition Mask Register (CMR) are used to implement the ESP's conditional execution mechanism (see section 4.2.)

3.3.1 ALU Operations

The ALU is capable of the 16 different arithmetic or logical operations described below. ALU operations are specified by the opcode field of the microinstruction, described in Section 4.1.2. The operations are listed here in order of their opcodes. A more complete description of each opcode follows the chart.

ALU Functions

Function	Description
ADD	Two's Compliment Addition with Overflow Protection (Clipping)
SUB	Two's Compliment Subtraction with Overflow Protection (Clipping)
ADDU	Two's Compliment Addition with no Overflow Protection
SUBU	Two's Compliment Subtraction with no Overflow Protection
CMP	A-B without result storage. Used to set Condition Codes
AND	Boolean AND operation
OR	Boolean OR operation
XOR	Boolean Exclusive OR operation
ABS	Absolute Value operation A=|B|
MOV	Move data from register A to register B. no CCR effect.
ASL2	Arithmetic Shift Left 2 with Overflow Protection
ASL8	Arithmetic Shift Left 8 with Overflow Protection
LS15	Left Shift 15 bits, clear sign bit, no Overflow Protection
DIFF	"Difference" Operation, A=$7FFFFF-B, with Overflow Protection
ASR	Arithmetic Shift Right, sign bit is preserved
END	End of the Microprogram

ADD is a classical saturating 2's complement addition operation used to create the sum of two signals. If the sum cannot be represented in 24 bits, full-scale positive ($7FFFFF) or full-scale negative ($800000) is substituted for the overflowed result. The operation performed is A = A + B.

SUB is a 2's complement classical saturating subtraction operation which yields the difference of two signals, analogous to ADD. It performs the operation A = A - B.

ADDU is an unsaturating addition operation. It behaves exactly as ADD, except that it lacks overflow correction. For this reason it is not normally used to add signals together. However, it is useful for generating ramp signals and for performing address arithmetic.

SUBU is an unsaturated subtraction operation, analogous to ADDU. Like SUB, it performs the operation A = A - B.

CMP is used to make signed comparisons between two values. It performs the operation A - B, but does not store the result back into A. The purpose of the comparison is to set bits in the Condition Code Register which can then be used to affect conditional instruction execution. The skip bit does not affect this instruction.

AND performs bit-wise logical AND-ing of each bit of two 24-bit operands. It can be used for many operations not usually thought of as signal processing, such as calculating power-of-2 moduli, performing lower-order bit truncation, or generating random numbers. It can also be useful for manipulating bits in the CCR and mask registers, and could conceivably be used to switch individual signal lines connected to devices in the ESP's external I/O address space.

OR performs bit-wise logical OR-ing of each bit of two 24-bit operands. Like AND, OR can be used for many operations not usually thought of as signal processing, as well as for manipulating the CCR and CCM bits.

XOR performs bit-wise logical EXCLUSIVE OR-ing of each bit in two 24-bit operands. Apart from the same types of logical applications mentioned for AND, OR, XOR can be used to produce the one's complement of a signal by XORing the signal with -1. The resulting 1's complement value is sufficiently close to the true arithmetic negative of the argument for most purposes. This is a more code-efficient method of negation than moving 0 into a register and subtracting a value from it.

ABS takes the absolute value of the argument, A = |B|. ABS performs the equivalent of a full-wave rectification on bipolar signals. This can be a very useful non-linear signal-processing function for deriving envelope signals and performing level detection. Negative arguments are negated with One's complement results.

MOV performs the basic data movement operation, MOVE B to A. The source and destination operands can be registers anywhere in the ESP's GPR memory space, or any SPR. MOV does not affect the status bits in the CCR. Thus moving a register back to itself is a true NOP.

ASL2 performs an arithmetic shift left of two positions on the operand, corresponding to a multiplication by four, i.e., A = B * 4. Saturation will occur if the sign bit changes during the shift operation. Zeros are shifted into the least significant two bits of the word. There is no unsaturating counterpart to this instruction.
Left shifting can be useful for normalizing a signal that is known to have been attenuated. ASL2 saves one cycle over the equivalent series of two ADD instructions. ASL2 can also be used as a logical left shift if bits 23 through 21 of the operand are the same.

ASL8 performs an arithmetic shift left of eight positions on the operand, corresponding to a multiplication by 256, i.e., A = B * 256. Saturation will occur if the sign bit changes during the shift. Zeros are shifted into the least significant eight bits of the word.
ASL8 can be useful for normalizing a signal known to have been severely attenuated. ASL8 is a considerable savings over the equivalent series of ADD instructions. ASL8 can also be used as a logical left shift if bits 23 through 15 of the operand are the same.

LS15 performs a left shift of 15 places on the operand, and clears the sign bit, equivalent to A = B * 32768. Consistent with the cleared sign bit, no saturation is performed on the result. The overall effect is to take the least significant eight bits and place them in the most significant positive part of the word. Zeros are placed in the low 15 bits of the word. The result is positive, having 8-bit precision.
LS15 is specifically intended to produce coefficients for interpolating between data values in adjacent RAM locations. The low 8 bits in a 24-bit address offset do not contribute to the physical addresses in a 64K word system (since addresses are left-justified) but can be thought of as representing the fractional part of the address. To perform a linear interpolation on RAM data at 2 adjacent memory locations, the difference between the 2 words can be multiplied by this fraction

and summed with the word at the lower address. For system memory configurations larger than 64K (see section 3.4.2), the fraction must be suitably masked and shifted farther to the left.

DIFF performs the "difference" operation A = $7FFFFF - B. DIFF is one instruction cycle more code-efficient than first MOVing the constant $7FFFFF into the A register and performing the equivalent subtraction.
DIFF is intended for generating crossfades between the outputs of delay lines. If B is the coefficient being applied to the output of one delay line, DIFF B,A creates the coefficient A to be applied to the other output to achieve an equal amplitude crossfade. With scaling, DIFF could also be used to generate difference addresses for coefficient look-up tables. The subtraction is unsaturated.

ASR performs a one bit position arithmetic right shift, equivalent to A = B/2. The sign of the operand is preserved in the result. ASR is intended for situations where simple divide-by-two's might be required in parallel with other Multiplier operations.

END is the ALU opcode which signals the end of the ESP microprogram. END is equivalent to one ALU instruction, but a multiply, or read or write of external RAM or I/O can be performed during the END instruction. During execution of the END instruction the following actions occur:

Host access is granted to GPR and microinstruction memory. (sec 5.2)
The Address Generator's Delay Base Address Counter is updated. (sec 3.4.4)

3.3.2 Condition Code Register

Associated with the ALU is a Condition Code Register (CCR) whose 6 bits reflect the state of previous ALU operations. The CCR is mapped into the ESP's operand address space as a special purpose register (SPR).

Condition Code Register Bits

Bit	23	22	21	20	19	18
Code	N	C	V	LT	Z	NOT

The CCR contains 5 flags which are derived in most ALU operations from basic arithmetic results.

Z zero result
C carry (or no-borrow)
N negative result
V result overflow
LT less than

LT is computed as LT = (V AND NOT N) OR (NOT V AND N). LT is necessary because no true combination of the other flags can determine a "less-than" condition. More details are available in Section 4.2.2.

3.3.3 Setting Condition Codes (Pipeline Considerations)

All ALU operations except MOV and END will set bits in the CCR based on their results (unless the operation occurred on a conditionally unexecuted step). Section 4.1.2 contains a complete description of the computation performed by each ALU operation. CCR bits become valid at the end of the complete ALU instruction cycle. Thus, they are available for reading by the next ALU operation, but not by the next Multiplier operation, just like a GPR (see Section 4.3.1 for details of the ESP execution cycle). CCR bits will be active for the conditional execution of both the ALU and Multiplier phases of the subsequent microinstruction cycle.

Condition Code Computation

	FLAGS				
ALU OPERATION	N	C	V	LT	Z
ADD,SUB	X	X	(1)	X	X
ADDU,SUBU	X	X	X	X	X
CMP,DIFF	X	X	X	X	X
AND,OR,XOR	X	-	-	-	X
ABS	0	(2)	-	-	-
ASL2,ASL8	X	-	(1)	-	X
LS15	0	(3)	-	-	-
ASR	X	(4)	-	-	-
MOV	-	-	-	-	-
END, NOP	-	-	-	-	-

(-) Indicates that this flag is not altered by the operation.
(1) V indicates whether saturation as opposed to overflow occurred.
(2) C receives the sign bit before the ABS operation.
(3) C receives the shifted bit that was cleared at the sign position.
(4) C receives the LS bit that was shifted right out of the operand.

Neither the MOV nor the END instructions affect condition code bits. (see Section 5.2 for a full description of END cycles).

The MOV instruction is a special case for two reasons:

1. ALU NOP instructions are actually assembled as a MOV instruction where the source and destination are the same GPR.
2. Since the CCR is a writeable register, you may move a value into it (after masking certain bits to force particular conditions, for instance). Preventing MOV from affecting CCR bits resolves any ambiguity as to the state of the CCR following the completion of the MOV operation.

3.4 Multiplier

The Multiplier multiplies a 24 bit multiplicand by a 24 bit multiplier and produces a 48 bit product during every instruction cycle. If the MAC bit in the Operand Select Field is set (see section 4.1.5), the product of the multiply will be added to the previous contents of the accumulator. This previous value may have been preloaded, or was the result of a previous multiply or accumulation.

24-bit operands for the Multiplier are fetched on the X and Y busses from GPRs specified by the C and D operands or from the DIL register and the GPR specified by D. The most significant 24 bits of the product appear on the Z bus and will be written back to the C operand address or possibly to the DOL FIFO. This corresponds to a "fractional" mode of multiplication.

The least significant 24 bits of the product are also available to the ALU in a SPR register, MACL. Thus the Multiplier can function as an integer multiplier as well as a fractional multiplier. In fact, both MACH and MACL, the upper and lower halves of the multiplier/accumulator, are SPRs registers and can be preloaded and recovered by any ALU operations.

The product will always be normalized, that is, shifted left, as either one or two bits prior to entering the accumulator, as determined by the Product Shift Mode bit in SIGREG. The reason for this is the following.

A normal 24 x 24 bit hardware multiplication will result in a 48 bit product with 2 sign bits. By left shifting the result by 1 bit, we can strip the extra sign bit and maintain true 24 bit two's complement data. The left shift by 2 operation is very useful as a range extension. This means that the normal +/-1 range system can be extended to a +/-2 system and by using the left shift by 2 function for the multiplier output the 2 extra sign bits that are created are effectively removed.

The Multiplier's accumulator contains two "guard" or extension bits which allows the intermediate partial sums in a summation of products to far exceed the ESP's normal range of values. As long as subsequent summations cause the accumulator to underflow back into the least significant 48 bits, the final result output from the multiplier will be an accurate representation of the sums of products. Otherwise, the Multiplier will substitute a saturated value for the overflowed result when it is output on the Z bus or read on the X or Y busses as SPRs.

If an accumulation at the current instruction is called for by a set MAC bit but the instruction is skipped through the conditional execution mechanism, no accumulation will take place. No output will result from the MAC, and its accumulator will retain the sum from the previous un-skipped step.

3.5 Address Generator

The address generator provides up to 24 bits of address for the RAM interface every instruction cycle. It has three main addressing modes which are selected by the 3 bit RAM Control field in the microinstruction (see section 4.1.6). The Address Generator can compute an address for one of the following operations on each instruction cycle:

table "A" reads and writes
table "B" reads
delay line reads and writes
external I/O reads and writes

The computed address comes out multiplexed in row and column form on the RAM data/address bus. The Address Generator uses the RAM Control field to drive the ESP's R/W pin and the RAS and CAS strobes. It also controls an I/O Space Select pin which is used to select between ordinary RAM accesses and accesses to other devices on the RAM bus.

3.5.1 Address Generator Architecture

The principle elements of the Address generator are:

A Delay Base Address Counter (**DBASE**)
A Delay Length register and comparator (**DLENGTH**)
Two Table Base registers (**ABASE** and **BBASE**)
A Memory size register to set delay increment (**MEMSIZ**)
The GPR's used as address offsets to memory (**OFFSET GPR**)
An offset adder

The DBASE Counter is the key to automatically maintaining signal delay lines in memory. All delay line accesses are performed relative to this recirculating decrementing counter, which configures RAM memory as a large circular buffer.

The DLENGTH register specifies how much of RAM should be devoted to delay line storage. Memory above the DLENGTH address will not be accessed by delay line read/writes as long as OFFSET GPR's are not greater than DLENGTH.

The Table Base registers ABASE and BBASE are used to provide base addresses for table-relative addressing of fixed data stored in RAM. The two registers allow two independent data areas to be established in memory.

The MEMSIZ register holds a power of 2 value used to define which address bits will used to address external RAM. The value written to this register is a function of the size of the external RAM in the system. It also sets the decrement value for the delay base counter (DBASE) and is a mask for delay address range comparisons.

The offset adder, the delay modulo comparator, and the delay base counter (DBASE) are the actual computational elements in the Address Generator. The method they use to compute physical addresses is detailed below.

The DBASE, DLENGTH, ABASE, BBASE, and MEMSIZ registers all appear in the GPR memory map as special purpose 20-bit registers which can be written to and read from by ESP microinstructions or the Host. Normally these will only be written once

during initialization procedures, but may be modified later, as well. (Note that there may be pipelining ramifications to changing these registers while programs are running.)

3.5.2 Selecting the Addressing range

The architecture of the Address Generator is based on the fact that all addresses in GPR registers are left-justified, a convention which has several advantages. One is that address bits to the right of the binary point serving as interpolation coefficients retain the maximum possible resolution. Another is that addresses can be stored directly in RAM without realignment to the left-justified 16-bit wide RAM data bus.

All Address Generator registers and data paths are 20-bits wide to accommodate up to 1 mega-word of RAM, this was done for future expansion. Because of the left-justified convention, the least significant physical address bit moves to the left within the ESP's standard 24-bit word when smaller amounts of memory are attached to the system. For example, if 16K words of RAM are used in a system, 14 address bits are needed. In this case, the least significant address bit would be (24-14=) bit 10. For 64K words, 16 address bits are needed, so the LSB is (24-16=) bit 8. The position of the least significant address bit is used by the Address Generator to determine how much to decrement the 20-bit wide DBASE pointer each sample period, and how many address bits to compare during delay address computation. The value in the MEMSIZ register is used for both computations.

The MEMSIZ register must be loaded with a 1 bit for each corresponding unused address bit. For instance in a 64K word system 16 address bits are used, so the leftmost 16 MEMSIZ bits should be 0 and the rest should be 1. In this case the MEMSIZ value would be $0000FF. In a 256K word system, 18 addresses are needed so $00003F would be used.

3.5.3 RAM Addressing modes

There are three basic RAM addressing modes which can be specified by the 3 bit RAM control field of each microinstruction. These are Delay Line, Table, and External I/O addressing. See Section 4.1.6 for the control fields for RAM addressing

Delay Line Addressing

This is delay counter-relative addressing, and is used to access signal delay lines which have been established in memory. To create a delay line, data is written at one address but read from an address higher in the memory space. Physical delay line addresses are generated by adding the relative address offsets fetched from GPR memory to the DBASE pointer. As the pointer decrements once each sample period, the read address eventually equals the address at which the write occurred. The distance in memory locations between the read and the write addresses equals the length of the delay line in sample periods. In the ESP, multiple delay lines are implemented starting at physical address 0 and extending up to and including the address contained in the DLENGTH register. Memory above the DLENGTH address is normally not accessed by delay lines, and can be used for tables A and B, or for other data.

Table Addressing

This is table base-relative addressing, and is used to access data stored at fixed locations in RAM. Physical table addresses are computed by adding relative address offsets (OFFSET GPR) to the contents of one of the two Table Base address registers. Positive or negative offsets can be used with the table registers, allowing them to define the high boundaries, low boundaries, or midpoints of data tables.

No restriction is placed on the contents of the Table Base registers, allowing data tables to be placed anywhere in the 20-bit RAM address space. The values in these registers are not masked by MEMSIZ. Tables should be placed above the address stored in the DLENGTH register if delay lines are used. Note that although only 2 table base registers are provided in hardware, appropriate reassignment of the table base registers by the ESP microprogram can provide an arbitrary number of tables.

External I/O Space addressing

This is the ESP's third basic addressing mode, used to access devices in an external I/O memory space. This memory area is distinguished from RAM memory space by the assertion of the I/O select pin instead of CAS during an otherwise ordinary memory cycle. It is intended that external hardware devices which require parallel data be interfaced to the ESP by placing them in I/O space. These devices, which might be ADCs, DACs, memory devices, or even other ESP chips, are assumed to be able to decode their own addresses from the ESP's multiplexed data/address bus with the help of the I/O Select pin. (see section 3.5.2) I/O space is always addressed absolutely, that is, the address in the offset GPR becomes the direct physical address multiplexed onto the ESP's data/address bus.

3.5.4 Physical Address Computation

At the beginning of each instruction cycle, a 24-bit word is fetched from GPR memory along with the microinstruction at that address. In delay line and table addressing modes this word represents the relative address offset (OFFSET GPR) from the appropriate ABASE, BBASE, or DBASE counter. In I/O Space addressing mode it represents the absolute address of a device in the external I/O Space. The physical address computation thus depends on the addressing mode specified by the RAM Control field in each microinstruction.

Delay Line Addressing Mode

This requires the most complex calculation of the three addressing modes. First, the offset is added to the DBASE counter. The sum is then compared to the DLENGTH register. If the sum of DBASE + offset is greater than DLENGTH, then DLENGTH is subtracted from the sum, and the resulting difference is used as the physical address. Otherwise, the sum is used as the physical address.

The following steps outline the process:

1. OFFSET GPR + DBASE counter = ASUM.
2. If ASUM ≤ DLENGTH then ASUM = RAM physical address
   else (ASUM - DLENGTH) modulo (DLENGTH) = RAM physical address.

The actual address presented to the RAM is masked by the MEMSIZ register.

In addition to the address calculation performed each instruction cycle, the DBASE counter must be updated once each sample period (microprogram iteration). During each END cycle the Base Address Counter is decremented

DBASE = DBASE - 1,

the equivalent of one RAM address as determined by MEMSIZ. When DBASE decrements past zero it is loaded with the value held in the DLENGTH register.

Note: the DBASE counter and DLENGTH register must be carefully initialized when the microprogram is loaded in order to work properly with data tables.

They must be set up before data tables are established to prevent the program's delay lines from overwriting table data. More details on initialization are covered in section 5.4.3.

Table addressing mode

Tables can access all of physical memory. This requires less computation than for delay line addressing, since the offset is added directly to the table base register specified in the RAM control field.

OFFSET GPR + ABASE (or BBASE) = RAM physical address

No boundary comparison or limiting is performed on the resulting address sum, and no overflow detection or correction is performed. Addresses are calculated identically for the ABASE and BBASE table base registers, even though the RAM control field does not allow for ESP writes to the Table B address space.

I/O addressing mode

I/O addressing actually requires no address computation at all, as the address offset fetched from GPR memory becomes the physical address in I/O Space.

3.6 Busses

3.6.1 Internal Operand Busses

The ESP contains three major data busses, labeled X, Y and Z. The X and Y buses are used to fetch operands for the ALU and the Multiplier and are 24 bits wide. The GPR specified as the ALU's A operand is always fetched on the X bus, and the Multiplier's C operand is always fetched on the X bus. Likewise, GPRs specified as the ALU's B operand or the Multiplier's D operand are always fetched on the Y bus. The Z ia a 24 bit wide bus used to deliver results back to the A and C GPRs and/or the DOL FIFO from the outputs of the ALU and the Multiplier.

Whether the Z bus writes back to the A and C GPRS and/or to the DOL FIFO depends on the "addressing mode" specified by the operand select field of the microinstruction. Operand select modes are covered in detail in section 4.1.3.

The Multiplier and ALU share the X, Y and Z busses by trading off their use on different phases of the microinstruction cycle. Bus cycle timing is covered in more detail in section 4.3.1.

All of the ESP's standard GPR register memory can be fetched equivalently over the X and Y busses. GPRs can be freely selected as A, B, C or D operands.

3.6.2 External RAM Busses

RAM Data/Address Bus

The RAM data/address bus is controlled by the ESP and allows access to both external memory (RAM) and to externally addressed devices. This bus consists of 16 bi-directional pins used to pass 16 bits of data between external RAM and the DIL/DOL registers, and 20 bits of address in 2 phases from the address ALU to external RAM. The 20 address bits are divided into row addresses and column addresses. These addresses can be used directly by dynamic RAM, or latched externally for static RAM. The state of this bus is determined by the RAM Bus Control Lines (below). The RAM data/ address bus is multiplexed as follows...

GPR bits to DRAM mapping

RAM Data/Address	ESP GPR Data Bit	RAM ROW Address	RAM COL Address
15	23	15	23
14	22	14	22
13	21	13	21
12	20	12	20
11	19	11	19
10	18	10	18
9	17	9	17
8	16	8	16
7	15	6	7
6	14	4	5
5	13		
4	12		
3	11		
2	10		
1	9		
0	8		

Note that as more RAM is added to the system the address bit are increased by appending to the lower portion of the bus. To understand the above chart a little further explanation is in order. The interface to DRAM is accomplished through the 16 lines labeled RAM Data/Address. These lines are multiplexed three ways:

1. Row Address
2. Column Address
3. Data Input or Output

The minimum DRAM expected to be used in a system is 64K. Therefore the address presented to the external DRAM is GPR data [23:8], with the lower 8 bits presented as the RAM row address. The 16 bit data is always returned to the ESP DIL left justified to bits [23:8]. As the amount of RAM is increased additional bits are appended to the row

and column addresses. These additional GPR bits [7:4] are distributed to the row and column lines as shown in the table.

RAM Bus Control Lines

RAS	active low row address strobe
GATE	active high column latch strobe
CAS	active low column address and data in strobe for RAM
IOSEL	active low data strobe for external devices (RAM, I/O, etc.)
DWEN	active low write enable

Memory vs. I/O Space

The ESP can swap its external data space between addressing RAM and addressing other external devices, such as I/O. These external devices can conceivably be anything from parallel CODEC interfaces to static memory devices to other ESP chips. Bus cycles intended to address the I/O space are distinguished from normal RAM cycles by the assertion of the IOSEL pin instead of the CAS pin. The RAS, GATE, and DWEN pins adhere to their normal timing. Timing of IOSEL for read and write is identical to a CAS read cycle. But CAS is asserted later for write cycles, and IOSEL is not.

Addresses in I/O space are calculated differently by the ESP's Address Generator than RAM delay line or data table addresses (see section 3.4.4). Address offsets fetched from GPRs essentially bypass the Address Generator adders and limiters and go directly out onto the address lines. Thus devices in I/O space must be accessed at their absolute physical addresses. Normal read and write timing considerations apply to I/O space devices however, since their data pass through the DIL and DOL FIFO just like ordinary RAM data (see 3.5.3).

3.5.3 RAM Data Interface

Data Input Latch (DIL)

Data read from RAM or external I/O space enters the ESP and is latched into a Special Purpose Register (SPR) called the Data Input Latch (DIL). The DIL is 16 bits wide, left justified, so programs accessing the DIL will see the least significant 8 bits as 0.

Data is strobed into the DIL based on the value of the RAM Control Field in the microinstruction word (see 4.1.6). Data latched into the DIL by a read operation will remain intact until another read operation is explicitly called for by this field. This latency allows an ESP assembler to make certain optimizations in the scheduling of reads and writes, especially when duplicate read operations are specified.

Data Output Latch (DOL) and FIFO

ALU or Multiplier instructions which use RAM delay lines, tables, or the external I/O space as a destination, send their results to a 2 word deep output FIFO known as the Data Output Latch (DOL). It's output is multiplexed onto the RAM bus during a write cycle. Like the DIL, it is also 16 bits wide and is left justified.

The DOL FIFO can hold up to 2 words of RAM write data. This was intended as a means for holding data until write cycles become available as read operations are normally considered to have priority over writes. Prioritizing read operations over writes is a valid approach because most musical signal processing algorithms require many more delay line reads than write operations. Read operands must be fetched and ready for use at particular points in the signal processing microprogram, whereas the great majority of write operations are insensitive to the specific time at which their data actually is stored in memory (provided that the write occurs sometime during the same sample period). This fact allows the ESP to hold off relatively infrequent write operations while more urgent read operations are tying up the RAM bus. The output FIFO allows up to 2 write data words to be buffered up while continuous read operations are occurring.

The DOL FIFO is written to by specifying it as an ALU or multiplier destination in a microinstruction. It is unloaded by specifying a RAM write or dump-FIFO operation in a microinstruction. Two DOL writes will fill both words. Two RAM writes will empty both words. Writing an empty FIFO to RAM will cause the last word written to the DOL to be written to RAM. Overstuffing the FIFO will cause the oldest data to be "bumped out", preserving the last two words written.

Microcode assemblers are capable of scheduling RAM reads and writes by assembling address offsets and RAM control fields at specific microcode steps (see 4.1.6). They may simulate the run-time execution of the microprogram to a certain degree to know when RAM cycles will be needed by read operations and when they will be free for writes. They should flag the programmer if the DOL FIFO is going to overflow at run-time from a combination of too many write operations and insufficient free RAM cycles between read operations.

Note: Halting of the microprogram (by execution of an END operation while HALT is asserted) will reset the FIFO pointers. This has the effect of emptying the FIFO, but a subsequent RAM write may not output the last word written to the FIFO. For this reason, DOL writes and RAM writes cannot span sample boundaries.

4. Operation

4.1 Microinstruction Word

The microinstruction word is 48 bits wide. Presently, only 45 bits are used. It is broken down into 4 eight-bit operand fields, a 4 bit ALU opcode field, a 4 bit operand select field, 2 single control bits, and a 3 bit RAM Control field. The three least significant bits are unused.

Microinstruction Word Format

Bit #	47 ⇒ 40	39 ⇒ 32	31 ⇒ 24	23 ⇒ 16	15 ⇒ 12	11 ⇒ 8	7	6	5 ⇒ 3	2 ⇒ 0
Func.	Mul Op D	Mult Op C	ALU Op B	ALU Op A	ALU Opcode	Operand Select	Skip	MAC	RAM Cntrl	Unsed (0's)

4.1.1 Operand fields

The operand fields comprising bits 47 through 16 hold the addresses of the GPRs or special registers specified as the A, B, C, and D operands used in ALU and Multiplier operations. Each address is a number between 0 and 255.

Addresses from 0 through 191 indicate memory locations occupied by normal 24-bit wide GPRs. Addresses greater than 191 indicate special locations or control registers. They are numbered in descending order beginning at address 255 to allow the GPR array to expand upward in future implementations. Section 3.1.1 contains a table summarizing these special locations.

4.1.2 ALU Opcode field

Microinstruction word bits 15 through 12 specify one of sixteen possible ALU operations, or opcodes. These operations are described in detail in section 3.2.1, and are summarized below with their opcodes.

ALU Operation Mnemonics and Opcodes

MNEMONIC	OPERANDS	OPCODE	OPERATION
ADD	B, A	0	A = A + B with saturation arithmetic *
SUB	B, A	1	A = A - B with saturation arithmetic
ADDU	B, A	2	A = A + B
SUBU	B, A	3	A = A - B
CMP	B, A	4	A - B (compare A to B)
AND	B, A	5	A = A and B (bitwise)
OR	B, A	6	A = A or B (bitwise)
XOR	B, A	7	A = A xor B (bitwise)
ABS	B, A	8	A = | B |
MOV	B, A	9	A = B
ASL2	B, A	A	A = B left shifted 2, saturated
ASL8	B, A	B	A = B left shifted 8, saturated
LS15	B, A	C	A = B left shifted 15, sign bit clear
DIFF	B, A	D	A = $7FFFFF - B
ASR	B, A	E	A = B right shifted 1; sign extended
END		F	PC = 0 restart program

Section 3.3.3 lists the bits in the Condition Code Register (CCR) which are affected by each of the above opcodes.

4.1.3 Operand Select Modes

ALU Operand/Bus Mapping

For ALU instructions not accessing the DOL FIFO, the GPR specified as the A operand is the implied destination register for the operation, and is the GPR written back into by the Z bus. Thus in 2-operand instructions like ADD or XOR, the A and B

operands are operated on and the results are placed in A. In 1-operand instructions (MOV, ABS, etc.), the result goes into A, but only the B register needs to be fetched. (Note that the A and B operand fields may contain identical addresses specifying the same physical register.)

The RAM Data Input Latch (DIL) also can be fetched over the X or Y busses. But since it is read-only, sharing the same address as the write-only MEMSIZ GPR, it must not be specified as a data destination. In 2-operand instructions, the DIL stands in for the A operand as one source, with the B GPR as the other. The result is written to the A operand, as usual. In this case the DIL must be fetched on the X bus (as A normally would be). In 1-operand instructions, however, the DIL stands in for the B operand, and thus must be fetched on the Y bus. Again, results are written back to A. The situation is summarized below.

ALU to Internal Bus Mapping

Instruction Type	**X BUS**	**Y BUS**	**Z BUS**
(i) 1-operand	***	B reg.	A reg.
(ii) 1-operand	***	DIL	A reg.
(iii) 2-operand	A reg.	B reg.	A reg.
(iv) 2-operand	DIL	B reg.	A reg.

This mapping provides the most regular bus/operand connections and at the same time makes best use of the operand fields in the microinstruction word. In particular, B inputs to the ALU are always fetched on the Y bus, and A results are always on X and Z (cases i, iii, and iv). Also, the provision for selecting DIL on either X or Y allows DIL to be used effectively in 1-operand instructions (ii) and in addition yields a very powerful 3-operand addressing mechanism (iv). NOTE: For hardware reasons, when assembling 1-operand DIL instructions (case ii), the assembler must put the DIL's GPR address into the B operand field of the instruction word. This is the only time that DIL must be treated as a GPR.

Multiplier Operand/Bus Mapping

Bus/operand mapping is much more straightforward when considering the multiplier. This is due to the fact that it always receives two operands, and that its operation is orthogonal with respect to its two 24-bit inputs.

Thus for the multiplier, the D register always comes in on the Y bus, with the X bus being swapped between the C operand and DIL for those operations that require it. The result is always sent back to the C operand on the Z bus. Using DIL as an operand offers a 3-operand capability in those situations that might require it.

Multiplier Bus Mapping

Multiplier Mapping	X BUS	Y BUS	Z BUS
(i) Non-DIL	C reg	D reg	C reg
(ii) Using DIL	DIL	D reg	C reg

Operand Select Field

The sources and destinations of the ALU and Multiplier operations are controlled by the operand and operand select fields of the instruction. The operand select field can be thought of as representing the operand addressing mode for a particular instruction cycle.

In general, the ALU operates on operands A and B and returns its result to A. The Multiplier operates on C and D and puts its results in C. These selections are overridden, however, by specifying the source and/or destination registers to be the DIL register and/or DOL FIFO which are the interfaces to external RAM. The various combinations of A, B, C, D, DIL and the DOL FIFO are encoded into the operand select field, as shown in the table below.

Operand Select Field Encoding

Operand Field	ALU Source	ALU Dest.	MUL Source	MUL DEST	Example Syntax for this Operand Combination	
0	A Reg	A Reg	C Reg	C Reg	ADD B,A >A	D X C >C
1	A Reg	A Reg	C Reg	FIFO	ADD B,A >A	D X C >Delay
2	A Reg	A Reg	C Reg	C/FIFO	ADD B,A >A	D X C >C,Delay
3	A Reg	A Reg	DIL	C Reg	ADD B,A >A	D X Delay >C
4	A Reg	A Reg	DIL	C/FIFO	ADD B,A >A	D X Delay >C,Delay
5	A Reg	FIFO	C Reg	C Reg	ADD B,A >Delay	D X C >C
6	A Reg	FIFO	DIL	C Reg	ADD B,A >Delay	D X Delay >C
7	A Reg	A/FIFO	C Reg	C Reg	ADD B,A >A,Delay	D X C >C
8	A Reg	A/FIFO	DIL	C Reg	ADD B,A >A,Delay	D X Delay >C
9	DIL	A Reg	C Reg	C Reg	ADD B,Delay >A	D X C >C
A	DIL	A Reg	C Reg	FIFO	ADD B,Delay >A	D X C >Delay
B	DIL	A Reg	C Reg	C/FIFO	ADD B,Delay >A	D X C >C,Delay
C	DIL	A Reg	DIL	C Reg	ADD B,Delay >A	D X Delay >C
D	DIL	A Reg	DIL	C/FIFO	ADD B,Delay >A,Delay	D X Delay >C,Delay
E	DIL	A/FIFO	C Reg	C Reg	ADD B,Delay >A,Delay	D X C >C
F	DIL	A/FIFO	DIL	C Reg	ADD B,Delay >A,Delay	D X Delay >C

4.1.4 'No output' Multiplier Destinations

In certain circumstances it can be advantageous to multiply the C and D operands and hold the result in the Multiplier's accumulator without overwriting the C register (because we'd like to preserve C's original value) by writing to a dummy RAM location (a waste of RAM bandwidth). The situation might be represented by

MUL C,D > MAC,

where MAC means that the result should just sit in the Multiplier's accumulator for use on a subsequent step. If an assembler encounters this situation it should generate an operand select code for a write to the DOL FIFO input (field value $1 or $A), but the RAM control flags that signal a RAM write (see Section 4.1.6) will not be generated.

Instead, a flag will be set to discard the FIFO data and preserve that RAM cycle for a read operation.

4.1.5 Skip and MAC bits

The Skip bit, bit 7, indicates whether the microinstruction is 'skippable', or eligible for conditional execution. See section 4.2 for complete details on flow control and the conditional execution of instructions.

The MAC bit, bit 6, controls whether the Multiplier should perform an accumulate as well as a multiply operation during this instruction cycle. The MAC bit is in effect an opcode for the multiplier.

4.1.6 RAM Control field

Three bits in the instruction word are used to control the RAM data and address busses. These busses can access I/O devices and other ESP chips as well as various sizes of RAM arrays.

The value of the RAM control field is closely tied to the operand select mode (see 4.1.3) since it is derived in part from the combination of sources and destinations required during an instruction cycle. In general the RAM control field is pipelined away into a different microinstruction word than the operand select field to which it corresponds. Details of the RAM pipelining and its effect on the pipelining of microinstruction fields can be found in Section 4.3.4.

Bit 3 in the RAM control field directly maps to the FIFO output strobe. When it is set, the FIFO outputs its next queued data word. Bit 5 and Bit 4 are memory select bits. These control RAM access by enabling CAS and the I/O select pin. The various combinations of the FIFO and memory select bits are decoded according to the following table.

RAM Control Field Encoding

RAM Adressing Mode	Control Field			Active ESP Pins		
Function	**B5**	**B4**	**B3**	**CAS**	**IOSEL**	**DWEN**
read delay line	0	0	0	X		
write delay line	0	0	1	X		X
read table A	0	1	0	X		
write table A	0	1	1	X		X
read delay/dump FIFO	1	0	1	X		
read table B	1	0	0	X		
read from I/O space	1	1	0		X	
write to I/O space	1	1	1		X	X

The read delay/dump FIFO mode provides a way of discarding a word in the DOL FIFO (see Section 4.1.4). Dummy writes to the FIFO are used when a result of the Multiplier is not desired except in the Multiplier's accumulator. Other than this state, the FIFO bit and the read/write pin would be the same function. If the FIFO is holding 2 output words, the first word written is discarded, leaving it holding only the last word

written. If the FIFO is holding only 1 word, that word is discarded. Dumping an empty FIFO has no effect.

4.2 Conditional Execution

4.2.1 Basic Conditional Mechanism

The ESP has a relatively simple mechanism that allows individual steps of its microcode to be conditionally executed, giving a modicum of execution flow control within signal processing microprograms. Any microinstruction step may be flagged by the programmer as 'skippable', meaning that it will not be executed when the ALU's Condition Code Register (CCR) is in a certain state. This state constitutes the skip condition, and is specified by loading a particular bit pattern into the Condition Mask Register (CMR). By loading various values into the CMR, instructions can be skipped according to a variety of different conditions. The conditions result from combinations of the 5 basic CCR flags N, Z, C, V and LT and their inverses. (See section 3.3.2).

When the Mask and Condition Code Registers correspond, an instruction with it's Skip bit set will execute as a NOP, or null operation. Both the ALU operations and the Multiplier operations specified by the skipped instruction become null operations. Skipping is accomplished by executing each phase of the instruction except writing of its final results to the specified target register, including the Multiplier's accumulator. This method of conditional execution maintains consistency both in the ESP's internal pipelines and in the overall timing of the system. The sample rate will remain constant regardless of the number of skipped instructions.

Instructions which are skippable cannot set the condition codes, regardless of whether they were executed or not. This restriction allows the mask register to be initialized and a condition tested (that is, have the CCR set by an instruction) at the beginning of a block of skippable instructions, with the condition being held true or false across the entire length of the block.

To detect a Skip condition during the instruction cycle of any skippable instruction step, the following logical computation is performed. The states of N, Z, V, C, and LT flags are logically ANDed with their associated mask bits and ORed together to yield a single SKIP condition. If the NOT bit is set in the mask register this SKIP condition is inverted. If the final result is TRUE (equal to 1) then the current instruction step is skipped. The step becomes a NOP by disabling the output of any results.

$$SKIP = NOT_m \oplus ((N \bullet N_m) + (Z \bullet Z_m) + (V \bullet V_m) + (C \bullet C_m) + (LT \bullet LT_m))$$

4.2.2 Arithmetic Condition Masking

There are 14 commonly used arithmetic conditions which can be tested based on the 5 CCR flags N, Z, C, V and LT. Seven of the conditions are simple combinations of these basic codes:

Positive Flag Test Conditions

Function	Test Flags
Equal (to 0)	Z
Negative	N
Overflowed	V
Lower than (unsigned)	C
Lower than or same (unsigned)	C OR Z
Less than (signed compare)	LT
Less than or equal (signed)	LT OR Z

Their negations result in the following additional seven:

Negative Flag Test Conditions

Function	Test Flags
NOT Equal	NOT(Z)
Positive	NOT(N)
Overflow clear	NOT(V)
Higher or same (unsigned)	NOT(C)
Higher than (unsigned)	NOT(C OR Z)
Greater than or equal (signed)	NOT(LT)
Greater than (signed compare)	NOT(LT OR Z)

The programmer loads a mask with 5 corresponding bits and a NOT bit into the Condition Code Mask register to select the condition which will cause instructions to be skipped.

The mask values for the 14 conditions are listed below. Note that these values are not copies of the bit patterns produced by the CCR when these conditions exist - they are merely the mask values that are necessary and sufficient for detecting the condition. The mask codes shown in these tables are set up in the **negative** sense, that is if the condition is met do not skip the subsequent instructions. To change the sense of the test, simply invert the sense of the NOT bit in the mask register.

Arithmetic Condition Code Masks

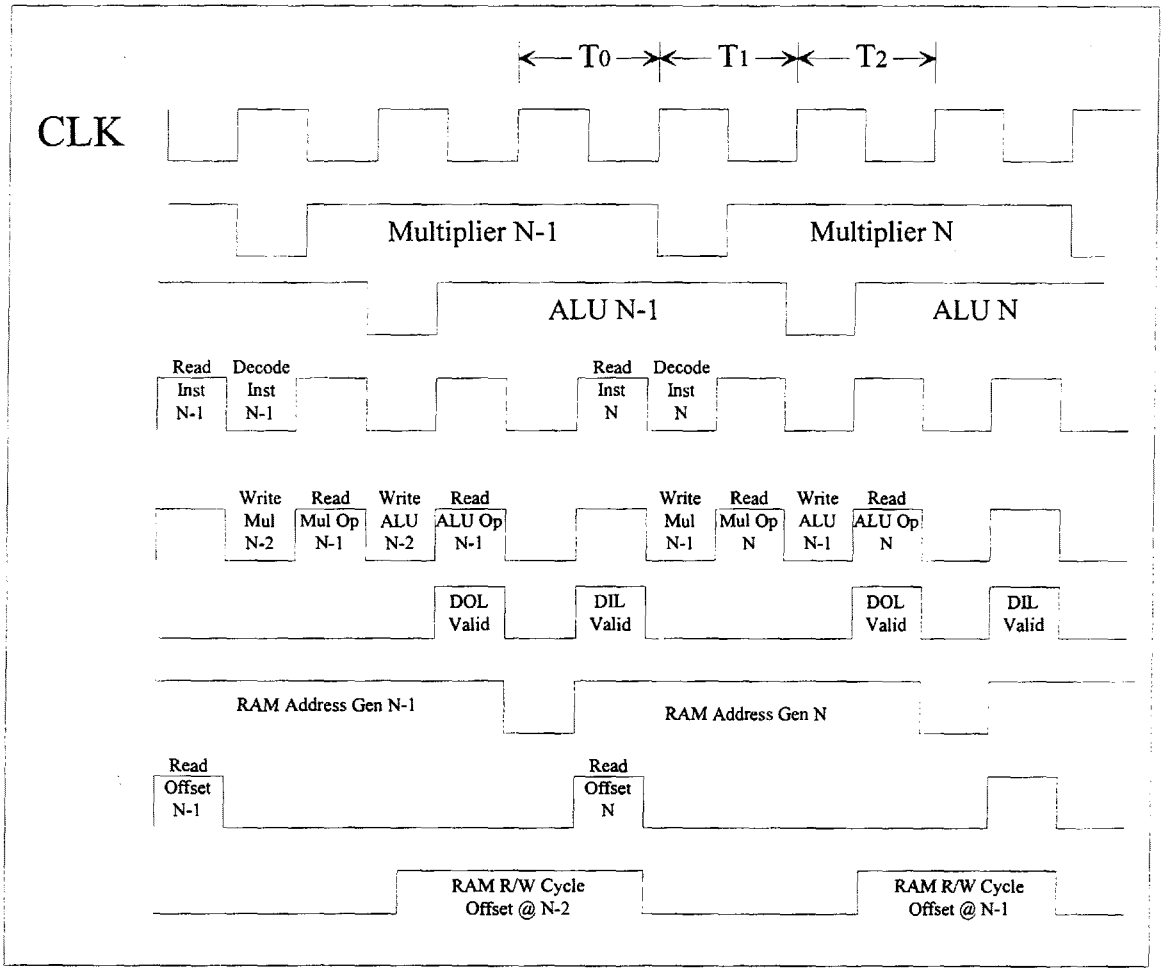
Condition	Mnemonic	N	C	V	LT	Z	NOT	Mask(hex)
Equal	EQ	0	0	0	0	1	1	0C
Not Equal	NEQ	0	0	0	0	1	0	08
Negative	NEG	1	0	0	0	0	1	84
Positive	POS	1	0	0	0	0	0	80
Overflow set	OV	0	0	1	0	0	1	24
Overflow clear	OVC	0	0	1	0	0	0	20
Lower than	LO	0	1	0	0	0	1	44
Higher or same	HS	0	1	0	0	0	0	40
Lower or same	LS	0	1	0	0	1	1	4C
Higher than	HI	0	1	0	0	1	0	48
Less than	LT	0	0	0	1	0	1	14
Greater or equal	GTE	0	0	0	1	0	0	10
Less or equal	LTE	0	0	0	1	1	1	1C
Greater than	GT	0	0	0	1	1	0	18
Always	TRUE	0	0	0	0	0	1	04
Never	FALSE	0	0	0	0	0	0	00

4.3 Microinstruction Execution Cycle

4.3.1 Basic execution cycle

The basic major execution cycle lasts for three input clock cycles, or T clocks. Each T cycle corresponds to a period of the external clock applied to the ESP. A T/2 clock is generated from the T clock for internal sequencing, and most internal events can be seen falling on these half-cycle divisions. A detailed diagram of the internal timing is shown below.

Internal Event Timing for the ESP



4.3.2 ALU/Multiplier Timing Offset

The timing diagram shows that the ALU, Multiplier and RAM Address Generator all have somewhat overlapping operating cycles. That their cycles are not completely coincident has important ramifications from a programming point of view. In particular, it can be seen that ALU operations lag Multiplier operations by one T cycle, 100ns. This lag is necessary because the Multiplier and ALU trade off their use of the X, Y and Z busses. The result is that the output of an ALU operation at any given microinstruction cycle is not available for the Multiplier to fetch as an operand during the next microinstruction cycle. Thus the programmer must leave one step between the generation of a value in the ALU and its use in the Multiplier.

Note however that because the multiplier writes its result before the ALU fetches its operands, Multiplier results can be used in the ALU on the very next step - no pipelining needs to be accounted for from a programming point of view.

4.3.3 Access to MACH and MACL

The Multiplier's 48-bit wide accumulator is also Special Purpose Register (SPR) pair MACH and MACL. ALU operations can use either of these two registers as source or destination operands. There are some non-obvious aspects to their successful use, however, due to pipelining considerations and to the fact that they are really part of the Multiplier and not independent registers. When the ALU reads MACH or MACL, the contents will be appropriately and conditionally saturated depending on the accumulators contents at the time of the read.

Preload and Postfetch

The instruction cycle timing diagram shows for the same microinstruction step N that Multiplier operations lead ALU operations by one T clock period. This means that ALU operations that preload either MAC half must be performed on the step *prior* to the one where the accumulation is to take place.

Note that ALU cycle N-1 completes sometime during the middle of Multiplier cycle N. It is at this point that the ALU result is loaded into the Multiplier to be accumulated at the end of cycle N.

Multiplied or accumulated results can be fetched from MACH or MACL by the ALU on the microinstruction step following the Multiplier operation. Note that fetching MACL with the ALU is the *only* way to gain access to the lower-order 24-bits of Multiplier results, since this half of the MAC is not normally enabled onto the Z bus at the end of a Multiplier operation.

MACH and MACL Volatility

Unlike standard GPR's, MACH and MACL are volatile. This means that they are recycled with fresh data every Multiplier cycle, regardless of whether a multiply or an accumulate took place. To preserve the contents of the accumulator between instruction steps not using the multiplier, the operation MAC = MAC + 0 x 0 is used. This is equivalent to a multiply/accumulator no-operation (NOP). This is also necessary when operands must be preloaded more than one cycle ahead, or results need to be retained in the accumulator past the current instruction cycle. This is a necessity, too, if it is desired to preload all 48 bits of the MAC, an operation requiring 2 ALU cycles.

When data is preloaded into the MAC, it is normally meant to be used as part of an accumulation during the next Multiplier cycle. If the next cycle specifies a *multiply* rather than a multiply/accumulate, however, that data will be lost. Instead, the accumulator will retain the result of the multiplication, as if no data had been preloaded. Thus the accumulator always receives the result of the previous cycle's operation.

The ESP assembler interprets the lack of any specified multiplication as a NOP for the Multiplier, and assembles the microinstruction as a multiply/accumulate of 0 times 0. This any data already in the accumulator will be retained across microinstruction cycles when no Multiplier operation is specifically stated. the programmer can explicitly include "0 X+ 0 > 0" to achieve a NOP.

MACL and MACH as Multiplier Operands

MACH and MACL can *not* be used as Multiplier source operands. The Multiplier performs its operand fetch 1/2 T cycle before the ALU writes the result of its operation at

the previous step. The destination of the previous multiply can be used, however, as a multiplier source operand.

Specifying MACL as a destination operand will cause the upper half of the accumulator to be duplicated in the lower half. The result of specifying MACH as a destination operand is undefined.

4.3.4 RAM Pipelining

One ESP access to RAM is guaranteed to be available each microinstruction cycle. In fact, the basic microinstruction period is tied to the relatively slow speed with which RAM accesses can occur. Because of slow DRAM access speed and the fact that the ESP takes a complete microinstruction cycle to calculate a complete physical RAM address, raw address offsets and RAM control fields must be supplied to the Address Generator considerably ahead of the time at which data associated with the address is needed or is ready. The RAM pipeline length, or the number of instruction cycles needed ahead of time, is 2. Looking at the RAM timing diagram in section 4.3.5, two complete T cycles can be seen to elapse between the fetch of the raw address offset for step N (OFFSET N) and the appearance of data corresponding to the resulting physical address on the multiplexed address/data bus (DATA N).

During the first cycle, a microinstruction and an offset are fetched and the physical RAM address is computed by the Address Generator (RAM ADDRESS GEN N) according to the instruction's RAM control field. This computation may involve an add, a comparison, and possibly a swap. (Section 3.4 gives more details on the operation of the Address Generator.)

During the second cycle, the row and column parts of the RAM address appear on the ESP's multiplexed data/address pins, suitably gated with the RAS, CAS, GATE and R/W signals. If RAM is accessed for a read operation, the data will be available to read during the latter half of this cycle and will be latched into the DIL by signals derived from the RAM control field. It will be available to be fetched as an ALU or Multiplier operand at the beginning of the next microinstruction cycle. If the RAM access is a write, the data output of a past ALU or Multiplier operation must be available from DOL FIFO to be transferred to the ESP data/address pins.

READ vs. WRITE Timing

An inconsistency in the pipelining scheme is caused by the fact that data being read from RAM needs to be available before the beginning of the cycle on which it is used, while data being written to RAM are not available until the very end of that cycle. That the execution of the Multiplier and ALU are skewed with respect to each other spreads this time window out even further, to four T cycles. The result is that addresses associated with reads at a particular step must be available earlier than addresses corresponding to write operations from that step.

In particular, read addresses must be pipelined back 2 microinstructions prior to the instruction cycle that they are needed on. This is dictated by the basic pipeline length of the ESP. Write addresses, on the other hand, are pipelined back zero cycles. Write data will not be ready until almost 2 microinstruction cycles have elapsed since the write operation's instruction was fetched. The 2 cycle pipeline length combined with the nearly-

2-cycle-late appearance of write data means that addresses for write data can be assembled at the same step as the instructions that write the data.

Write Data Latency

The different pipeline amounts for read and write addresses creates a situation of potential conflict between reads at one step and writes 2 steps later. Both operations want to see their RAM addresses at the same microinstruction address, even though they are occurring on different steps. This conflict is ameliorated to a great degree by the latency of data written to the DOL and the fact that the DOL contains a 2 stage FIFO.

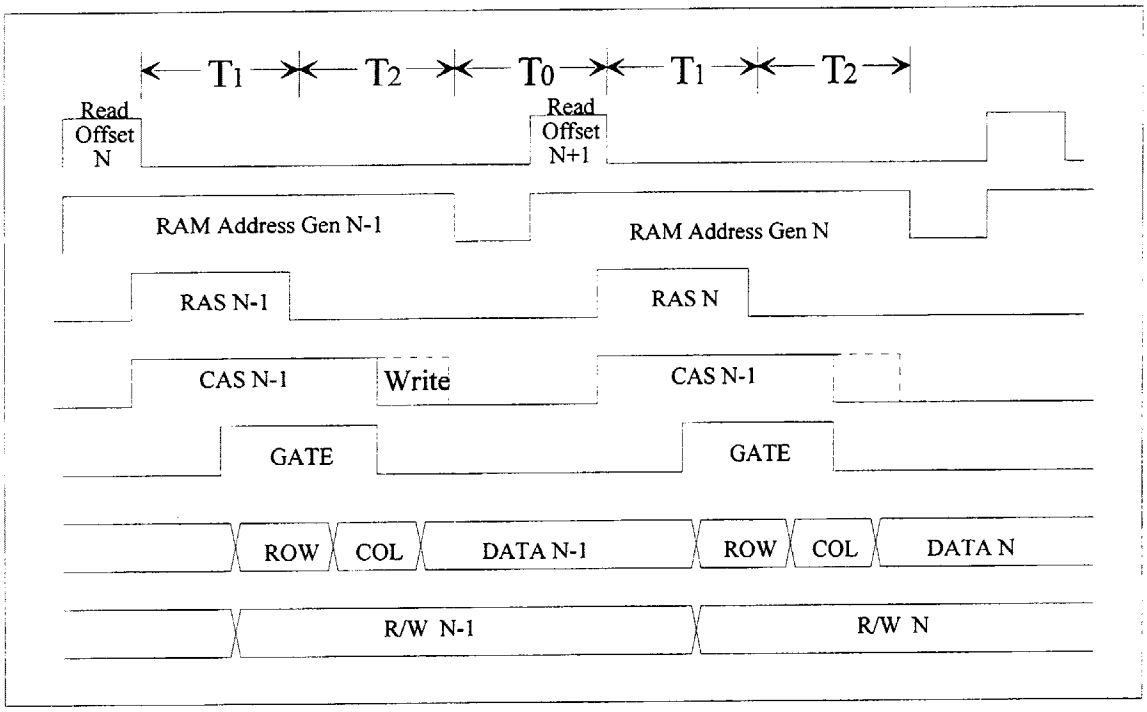
The actual operation of the DOL FIFO is covered in greater detail in section 3.5.3, but its purpose is to store pending write data until such time as no read operation is in progress. Because of the FIFO, up to 2 data words can be buffered and waiting for a free RAM cycle. When an available cycle arrives, the write address for the (first) waiting data word will have been computed by the Address Generator, and the output of the FIFO can be output to the data/address pins.

RAM Address Assembly

It is the responsibility of an ESP assembler to simulate the scheduling of the RAM interface and DOL FIFO to determine where to assemble address offsets. Write addresses must be positioned so that they match the write data that will be pending in the DOL FIFO at runtime. The assembler also must warn the programmer if read/write conflicts will occur, that is, if insufficient free RAM cycles to handle the number of specified write operations will cause the DOL FIFO to overflow. The scheduling is somewhat intricate and is complicated by the facts that A) the ESP sometimes "throws out" unwanted data in the DOL FIFO and B) data read to the DIL is latent through intervening write operations.

The RAM Control field of the microinstruction must be pipelined along with the address it corresponds to. The 3 bits in the field are needed to control the ESP's RAM interface at the time the address and data are written to or read from external memory, not at the time the RAM data is output or fetched by the ALU or Multiplier. In fact, the control bits must travel through a 1 microcycle delay pipeline register so that they remain synchronized with the address data as it passes through the 1 cycle delay caused by the Address Generator.

DRAM Timing



CAS does not fall when I/O SEL is active. I/O SEL timing is the same as CAS but is not stretched on a write cycle.

5. Host Interfacing

5.1 Host Interface Registers

The ESP/Host interface allows the ESP to be addressed as a peripheral to a Host microprocessor. The interface is comprised of 8 pins which carry the multiplexed data and address of the desired interface register, plus R/W, chip select and address strobe pins.

The 8 address lines map the ESP into a 256 byte block of the Host's address space. This block contains registers which allow the Host to read and write to all of the memory internal to the ESP as well as into the I/O and external RAM memory spaces accessed via the ESP's external data bus.

5.1.1 Host Interface Memory Map

Hex Address Register name

Add. (hex)	Register Name	Bits	Description / Function	
00	GPR	23-16	GPR / SPR Data MS-byte	
01	GPR	15-8	GPR / SPR Data	
02	GPR	7-0	GPR / SPR Data LS-byte	
03	INSTR	47-40	Mult D Operand Address	
04	INSTR	39-32	Mult C Operand Address	
05	INSTR	31-24	ALU B Operand Address	
06	INSTR	23-16	ALU A Operand Address	
07	INSTR	15-8	ALU Opcode / Operand Select	
08	INSTR	7-3	Skip Flag + MAC + RAM Control	
09	DIL	23-16	DRAM Input Latch MS-byte	
0A	DIL	15-8	DRAM Input Latch	
0B	DIL (always read as 00)	7-0	DRAM Input Latch LS-byte	
0C	DOL	23-16	DRAM Output Latch MS-byte	
0D	DOL	15-8	DRAM Output Latch	
0E	DOL (always read as FF)	7-0	DRAM Output Latch LS-byte	
0F	DADR (write last to transfer data)	23-16	DRAM Address MS-byte	
10	DADR	15-8	DRAM Address	
11	DADR	7-4	DRAM Address LS-byte	
12	Host Control	2-0	ESP State / Control Flags	
14	RAM Access Control	7-6	RAM,I/O, Read/Write Select	
16	Program Counter		For Test Purposes Only	
17	Internal Refresh counter		For Test Purposes Only	
18	Host serial control	7-0	Serial I/O Format and Control	
1F	Halt enable (write only)		Stop ESP Execution	
	Frame Counter (read only)		For Test Purposes Only	
80	Read Select: GPR + INSTR	7-0	Read Address	Address Ranges:
A0	Write Select: GPR	7-0	Write Address	GPR $00 - $BF
C0	Write Select: INSTR	7-0	Write Address	SPR $EA - $FF
E0	Write Select: GPR+INSTR	7-0	Write Address	INSTR $00 - $9F

5.1.2 GPR/Instruction Access

The internal data busses of the ESP are constantly being accessed when microprograms are executing. Thus one microinstruction cycle per sample period is

dedicated to Host/ESP transfers. This time, during the execution of the END statement, is the only time during program execution a transfer may take place between the Host interface registers and the internal GPR and instruction memory (see 5.2). The transfer takes place in lieu of the fetch of A and B ALU operands. Since only one transfer is allowed per sample period, a protocol exists which should be invoked to read or write internal data.

Host Access OK/

When accessing internal GPRs or microinstructions, the Host must always check that the Host Access OK/ bit in the Host Control register is low or set to 0. Reading or writing data or addresses into the GPR or Instruction registers can upset data transfers that are already in progress if the Host OK bit is set.

To read data, the Host must check Host Access OK/ and then request the data by writing a GPR number in the GPR READ register. The OK bit will be set and a copy of the register and instruction field data stored at that address will be placed in the Host access registers at the execution of the next END statement. When this data is valid, the Host Access OK/ bit will go low again, and the data can be read by the Host.

To write data, the Host must check Host Access OK/, and then place the data in the GPR and/or instruction field. Then the address of the GPR or instruction is written into one of three different write registers, depending on whether just GPR data, INSTR word data, or both get transferred. Writing the destination address at one of these locations causes Host Access OK/ to be set, and tells the ESP to transfer the new data to the internal memory on the next transfer slot (END statement). The OK bit will go low again when the ESP has completed the data transfer to the internal memory, a maximum of one sample period later.

This method of transferring data to the ESP is meant to be used as a way of modifying the parameters of a program in real time, without actually downloading a whole new algorithm. Examples of this are real time changes in reverb mix or chorus frequency. When downloading an entire new program, the CPU will hold the HALT pin high and stop ESP execution. This will allow higher transfer throughput to download the new algorithm quickly.

Note that the GPR and instruction fields and the address select registers will retain their last contents after a write has been completed. Thus the following technique could be used to, for instance, update only the Skip bit of a microinstruction:

1. The Host reads the microinstruction by writing the instruction address to Read Select at interface register $80, then checks the Host Control register for Host Access OK/.
2. Force HI the upper bit of the INSTR byte interface register ($08).
3. Copy the address back out of Read Select into INSTR Write Select ($C0) to initiate the write-back of the altered microinstruction.

5.1.3 RAM Registers

This group of registers comprises the 3 byte wide DOL, 3 byte wide DIL, the 3 byte DADR RAM address registers, and a RAM Access Control register.

With these registers, the Host may gain direct access to the RAM memory or I/O space for the purposes of preloading lookup tables, accessing sampled data, performing memory tests, and the like. To access RAM directly through these means, the ESP must be in its Halt state, caused by holding the HALT pin high. At other times, while the ESP is running, Host access to RAM memory must be provided for explicitly by dedicated microprogram instructions that read from memory into a designated GPR. (See section 5.4.1 for more details on the interaction between the Halt state, RAM Clear procedure and direct RAM access.)

The DIL and DOL Host registers are connected directly to RAM data through the multiplexed RAM address/data bus. During HALT, the FIFO pointers are reset, and the host interface can access the DOL directly. Space for DIL and DOL LO registers was reserved in the host memory map, but since the DIL and the DOL are both 16 bits wide, left justified, the low bytes will read back as zero.

RAM Addresses placed in the DADR registers bypass the ESP's Address Generator entirely, and become the physical RAM address which is delivered to the external memory devices. Addresses are always left-justified in the DADR registers. The number of address bits which are active is determined by the amount of RAM actually connected to the ESP. In an ESP configuration with 64K of RAM, for instance, the value placed in the DADR LO Host register is irrelevant for the purposes of addressing RAM. In any case, the most significant address register, DADR HI, will always be active because of the left justification of RAM addresses.

RAM Access Control

The RAM Access Control register is used to specify whether read or write operations are to be performed, and whether access should be made to RAM memory or the ESP's I/O address space. Its active bits are as shown below.

RAM Access Control Register

Bit	7	6	5 4 3 2 1 0
Function	Read Select = 1 Write Select = 0	RAM Select = 1 I/O Select = 0	1 1 1 1 1 1

Once the access mode is set by writing into this register from the Host, write operations to DADR HI are used to actually initiate the transfers of data to or from memory. **Note:** The ESP should be in HALT mode when accessing this register.

Reading RAM data

To read data, the RAM Access register is initialized for reads and the one or two least significant bytes of RAM address bytes are written to the DADR MID and (optionally) DADR LO Host registers. Finally, the most significant address byte is written to DADR HI. RAM data will become valid in the DIL Host register no later than 6 clock cycles afterwards.

Writing RAM data

To write data, the RAM Access register should be initialized for writes, after which the data to be written can be placed in the DOL Host registers. Then the least significant parts of the address should be placed in the DADR MID and LO registers, as above. Finally, the most significant byte is written to the DADR HI register to initiate the transfer. The write operation can be assumed to have been completed after a maximum of 6 clock cycles.

5.1.4 Host Control Register

The Host Control Register is a byte-wide register with control and status bits whose functions are shown in the table below.

Host Control Register

Bit	7 6 5 4 3	2	1	0
Function	N.A.	Host Access OK/ (read only, low active)	RAM Clear	Refresh Disable

The Host Signal Control register is concerned with Host access to the ESP's internal memory and has bits for controlling RAM clearing and RAM refresh disabling. Refresh Disable must be 0 when activating RAM Clear. Bits 1 and 2 are self-clearing. Bits 0 and 1 are useful only during Halt Mode.

Host Access OK/

This read only bit is used by the ESP to indicate when the information in the GPR and Instruction Host Interface Registers is valid. It also indicates that the ESP is currently busy completing a read or write operation. The Host should not disturb the ESP by writing data or addresses into its GPR/INSTR interface registers when Host Access OK is true. Note that host interfaced registers not pertaining to GPR and INSTR access can be read or written at any time without regard to Host Access OK/.

More details on Host access to the GPR and Instruction memory can be found in section 5.1.2.

RAM Clear

When the ESP is in its Halted state, setting this bit causes the RAM clearing process to be initiated. Prior to doing this, make sure that MEMSIZ and DLENGTH are initialized to correctly indicate RAM size. When zeros have been written to all of the memory specified, this bit will automatically clear. The Host may poll it to check the status of the clearing operation.

Refresh Disable

When the ESP enters its Halted state, it will normally continue to refresh the contents of external RAM. This is desirable for preserving the contents of dynamic RAMs. However, RAM refreshing (and clearing) is accomplished by counting through the delay memory address space using the Address Generator's DBASE counter. Under some circumstances, however, it may be undesirable to alter the contents of the Base Address Pointer while the

ESP is halted. The best examples of this are when using HALT as a means for synchronizing or using HALT to set the sample rate. While halted, waiting for the next sample to begin, we do not want the Base Address Pointer to commence RAM refresh, as this will disrupt delay lines. This bit stops DBASE from decrementing while in Halt Mode. Options for DRAM refresh are discussed in Section 5.3.1

5.1.5 PC, Internal Refresh Counter

These registers are intended for IC test purposes only. They should not be written to by ordinary ESP applications programs.

The Program Counter

The Program Counter (PC) is the microprogram instruction counter, which increments through the ESP's program memory while the chip is running. It will start at 00 and increment until an END instruction is encountered. The PC will then reset to 0 and begin execution if the HALT conditions are negative.

The Refresh Counter

This is a counter that is used to refresh the array of internal dynamic memory used to implement the ESP's GPR and microinstruction address space. Frame (bit 7) is a bit that is toggled when the counter completes refreshing the GPR or INSTR array. This bit therefore indicates which array is currently being refreshed.

5.1.6 Halt Enable Register

When the HALT pin is high, a write to this register forces the ESP to halt immediately, without waiting for an END statement. Normally, if the HALT pin is high the ESP will continue to execute instructions until an END instruction is encountered. Enabling this condition is meant to be used on power-up when the contents of microprogram memory are unknown or if a fault condition occurs and the host CPU wishes to stop execution immediately. If the ESP is executing a known program it is not recommended to use this feature, since the state of the machine could be unknown. For example it might be wise to ramp the serial output registers to zero before loading a new program, so that output D/A does not make a click. If the host CPU asserts the HALT pin to a high, the ESP will stop correctly when the END instruction occurs. The host CPU can now download a new program.

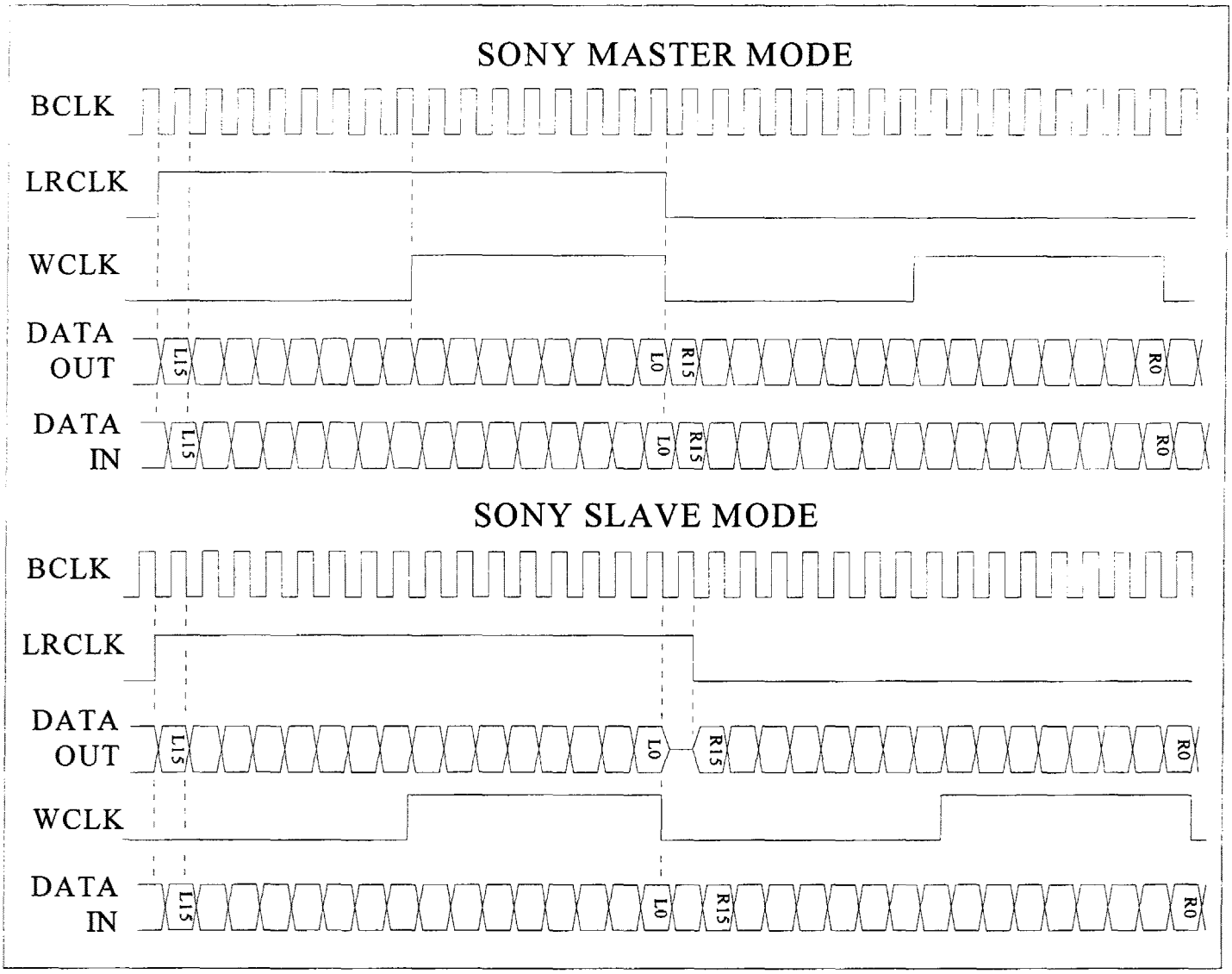
5.1.7 Host Serial Control Register

The Host Serial Control Register is used to setup the various modes for the Serial Clock Lines (LRCLK,WCLK,BCLK) and the Serial I/O Lines (SER[3:0]). The following chart indicates how the bits in this register control the mode of the lines. Master mode means that all Serial Clock lines are enabled. In slave modes these lines need to be driven from an external source.

Host Serial Control Register

Bit #	7	6	5	4	3	2	1	0
Set (1)	MASTER	SONY	SER3 out	SER2 out	SER1 out	SER0 out	1	1
Reset(0)	SLAVE	I2S	SER3 in	SER2 in	SER1 in	SER0 in	-	-

The ESP has four stereo 16-bit serial data lines which operate in conjunction with the bit clock, word clock, and Left/Right clock. If the ESP sources these signals it is in Master Mode. Each serial line can be configured as an input or an output. The timing diagrams that follow show the relationship on the control clocks and data for the various modes available. For more complete information on the timing of the signals refer to the detailed timing information in the end of the specification.



Sony Master Mode

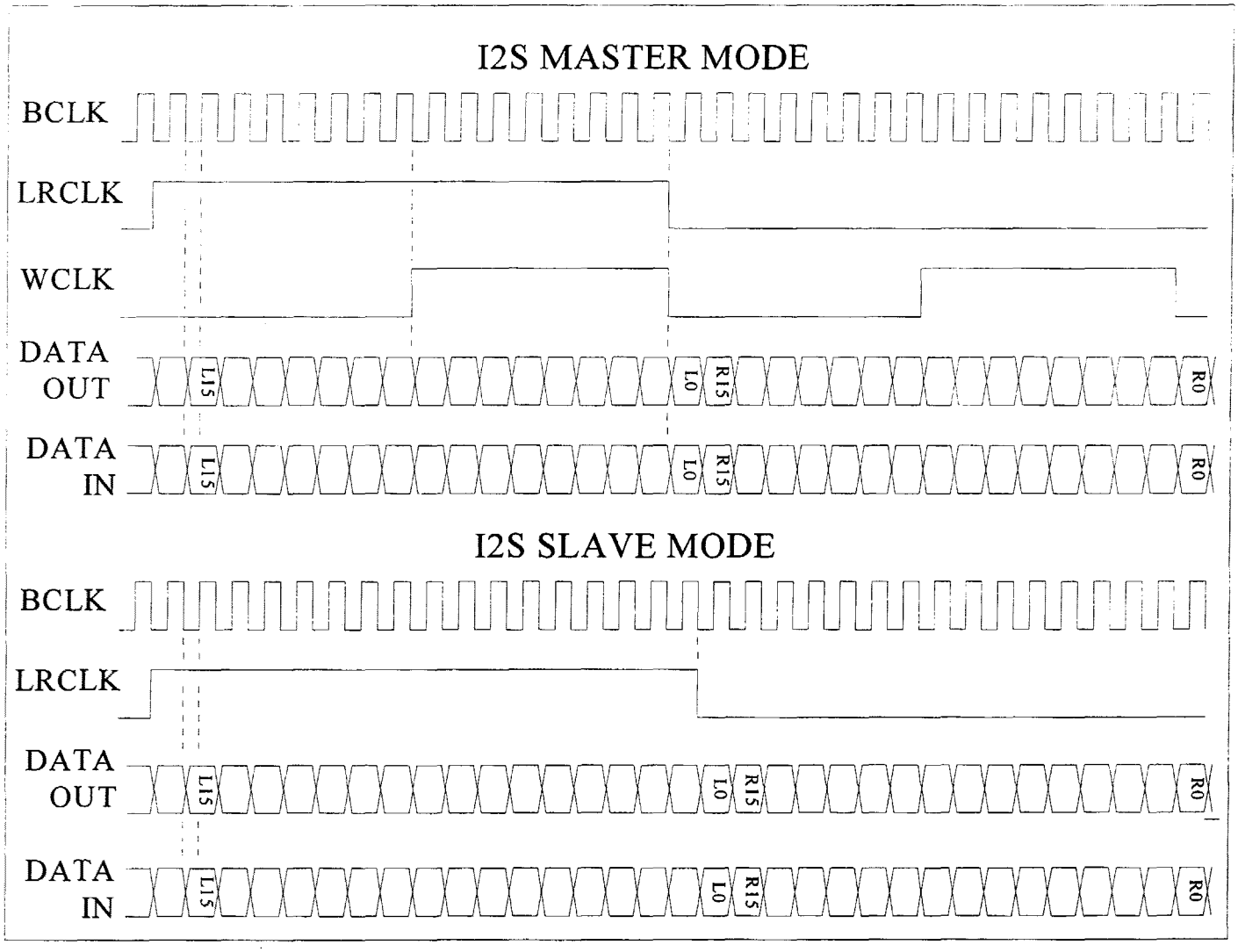
In Sony Master mode DATA OUT is changed after the fall of BLCK. DATA IN is latched on the falling edge of BCLK. LRCLK and WCLK are output immediately after the fall of BCLK.

Sony Slave Mode

The LRCLK is sampled when BCLK is low. If a transition of the LRCLK is detected then DATA OUT begins during the same BCLK low phase where the LRCLK change is detected. LRCLK **must only change** when BCLK is low. Note: The setup time to BCLK rising is very important.

DATA IN is justified with respect to WCLK. This means that the 16 bits shifted in prior to the fall of WCLK while the LRCLK is stable are used as the DATA IN.

Since DATA OUT responds to the LRCLK and DATA IN is justified with respect to WCLK the serial left and right data I/O transfers can occur at different times in the sample period.



In both I2S mode WCLK is not used for data transfers. The clock is made available in Master mode simple because it is there. In slave mode, DATA IN is the first 16 bits shifted into the Serial Port after a change in LRCLK. There is a one cycle delay after a change in LRCLK before the MSB is available. It is not necessary to have left serial data followed immediately by right serial data. LRCLK can remain high after the 16 left channel bits have been received. Right serial data will not be begin being received until LRCLK falls.

5.2 Host Interaction and the END statement

5.2.1 ESP End Statement

When the program encounters an END instruction the following actions are performed:

1. Resets program counter. This causes execution to begin at the start of the program
2. Enables halt line sampling. When the end instruction is encountered the HALT line is sampled. If the halt line is high, execution is suspended until the halt line is reset low. Once the halt line is low, execution resumes at the beginning of the program.
3. Triggers serial clocks. This starts the clocking sequence for serial data communication. In Master mode LRCLK will go high to begin the sequence. In Slave mode, when the LRCLK pin goes high data transfers will begin.
4. Decrements Delay counter DBASE. As discussed in Section 3.5
5. Allows host access. A single instruction or GPR location is available for access without halting program execution

5.3 Host Interface Pins

5.3.1 HALT pin

This input pin is used to place one or more ESP processors in the Halt state and to resynchronize them. Halt state is useful for initializing an ESP, allowing high-bandwidth access from the Host. (See section 5.4.1 for a detailed description of the Halt state.) Driving the HALT line high causes an ESP to restart its internal microprogram counter at location 0. HALT can be used to perform external sample-rate synchronization, single-sample step execution in a debugging environment, or program execution synchronization in a multi-ESP system configuration.

HALT is a level sensitive input. Holding it high causes the ESP to enter its Halt state at the next execution of the END instruction. The level of the HALT is sampled only while halted or during execution of the END statement, so the ESP will continue executing until the end of its microprogram before Halting. If RAM refresh is disabled by the Refresh Disable bit in the Host control register, this leaves the entire state of the ESP in a known condition when Halted. When HALT goes low, the ESP will continue executing from location 0.

If multiple ESPs are connected to the same HALT line, pulling the line low will cause all of them to begin executing their programs together. The microcode for all ESPs do not need to be exactly the same length, as HALT will be setting the actual sample rate. However, if delay lines are used by any ESP, RAM refresh must be disabled to prevent DBASE from commencing RAM refresh while halted.

Sample-Rate Sync

HALT can be used to synchronize an ESP to an external sample rate clock such as from an analog sub-system or from some other signal source. The first requirement for this method is that the microprogram execution time be equal to or shorter than the shortest sample period. The second is that RAM refresh must be disabled unless the sample period is never more than 1 instruction cycle longer than the microprogram. (see single cycle stretching, below.) The HALT pin is driven with an external sample clock that has been conditioned to provide a low-going "begin-sample" pulse that is shorter than the microprogram execution time.

When the "begin-sample" pulse goes low, the ESP begins executing its microprogram. Sometime during program execution, the "begin-sample" pulse will rise, causing HALT to go high. When the ESP encounters an END instruction, it will enter Halt state and wait for another "begin-sample" pulse before executing the next iteration of its microprogram.

For sample-rate synchronization to work properly when delay lines are used, RAM refresh must be disabled with the Refresh Disable bit in the Host Control register. This prevents the Base Address Pointer from being used for RAM refresh. As a general rule, microprograms using delay lines typically access enough RAM locations in a short enough time to satisfy the refresh requirements of dynamic RAMs. Therefore, disabling refresh under these conditions is of no serious consequence. Programs without delay lines should not disable the refresh feature.

Single-Sample Execution

Single-sample execution can be seen as a special case of sample-rate sync, where the "begin-sample" pulse occurs at a very low and/or irregular rate. The "begin-sample" pulse should be narrow relative to the execution time of the program. When the ESP has finished one iteration of the program, it will enter the Halt state and wait for another "begin-sample" pulse.

While Halted, the Host has full internal access to the ESP and may examine and/or change any ESP microcode, GPRs or special registers. Though RAM data may be lost if refreshing is disabled while Halted, refreshing RAM causes the Base Address (DBASE) pointer to be altered and lose track of its delay lines, an equally undesirable side effect. A solution is to perform single-sample execution in the following manner:

1. While the ESP is running, disable RAM refreshing using the Refresh Disable bit.
2. Use Resync to Halt the ESP when it is desired to suspend execution.
3. Immediately save the current Base Address pointer value.
4. Enable RAM refresh by clearing the Refresh Disable bit.

5. Perform any internal examination of the ESP required for debugging
6. Disable RAM refresh.
7. Reload the Base Address counter with the saved value.
8. Bring HALT low to resume execution.

5.3.2 Host Bus

The Host Bus is a bidirectional multiplexed address input and data I/O bus. The 8 bit host address specifies which ESP register is to be accessed by the host CPU. This 8 bit bus is latched into the ESP internal address latch upon the fall of the AS line. After the fall of the AS strobe, the data to written is placed onto the bus, the data will be latched on the rising edge of CS. In a read cycle, the ESP will place data on the bus to be read by the CPU. Data will remain on the bus until CS goes high.

5.3.3 Address Strobe

This pin is used to latch the input address from the host CPU. Once a stable address is available on the Host Bus, holding this line low will hold the address in the ESP internal address latch. This line is level sensitive, not edge sensitive, therefore it must be low during the entire access cycle after the address is latched. Any glitches on this line sould result in incorrect address information being written to the ESP. In normal designs the AS line and CS line can be tied together.

5.3.4 Chip Select

This pin is used to initiate data transfer to the CPU after the address has been latched. A low condition on the CS line indicates the ESP should find an open internal time slot and transfer data from the selected register to the Host Bus. Since the address is latched on the falling edge of AS and a low on CS initiates a data transfer, these lines can be tied together in most designs.

5.4 Host Initialization

5.4.1 Halt State

The ESP enters its Halt state when it detects a high condition on the HALT pin during the execution of an END statement. The HALT line is only sampled during the END cycle, so the ESP will not halt until it finishes executing a complete iteration of the microprogram.

When HALT is high, the ESP will perform its normal DBASE update (decrement, test, and possibly reload) before it halts. The microprogram counter, however, will not be reset to 0 until after it comes out of the Halt state, since it is being used for internal refreshing while the ESP is Halted.

While the ESP is Halted, the Host will be allowed more frequent access to internal GPRs and microcode - approximately one access per microinstruction period, or 300ns. Thus the Host will not normally need to poll the Host Access OK bit to determine when reading and writing can take place, since this period is shorter than a typical Host's

microprocessor instruction cycle. GPRs and microcode memory will appear to be static to the Host since they are refreshed automatically and transparently during Halt by the ESP.

External dynamic RAM, on the other hand, is not necessarily refreshed by the ESP during Halt. Refreshing may be disabled during Halt by the Host in order to preserve the state of the DBASE counter. This is necessary in situations where the HALT pin and Halt state are being used to synchronize the ESP to an external sample clock or where single-sample program execution is being performed for debugging purposes.

Note: GPR and INSTR Select Registers will inhibit GPR Refresh when the last GPR or INSTR addressed by the Select Register ($80, $A0, $C0, $E0). If the Select Registers contains a valid GPR (not SPR) address, that GPR will not be refreshed. This was necessary because host GPR/INSTR accesses are substituted for refresh cycles. If this condition persists for an extended period of time, that GPR may lose its data. When you are finished GPR and INSTR accesses, park the select register at a non-GPR address (like $FF) to avoid this.

Enabling RAM Refresh

Two bits in the Host Control register determine whether refreshing and/or clearing of RAM will occur during the Halt state (see section 5.1.5). When the Refresh Disable bit and RAM Clear bits are false, the ESP will begin refreshing RAM as after it enters the Halt state.

Refresh is accomplished by sequencing the DBASE counter through consecutive RAM addresses. A new refresh address will be generated every microinstruction period. Refreshing will begin at the address contained in the DBASE pointer, looping from 0 to DLENGTH, decrementing one RAM location each instruction cycle as determined by MEMSIZ. Thus the amount of RAM refreshed will correspond to the size of delay memory.

RAM Clearing

When new programs are loaded into the ESP, it is usually desirable to clear all of delay memory so that no residual noise wil exist in delay lines defined. To implement this, the ESP can be used to write zeros to its delay memory space while in the Halt state.

Setting the RAM Clear bit in the Host Initialization register will cause the ESP to begin clearing RAM. The ESP must be halted to do this. The DBASE pointer is initialized to the contents of the DLENGTH register. It begins counting down one RAM location at a time, as determined by MEMSIZ. Thus only the memory configured for delay lines is cleared by RAM Clear. To clear all of RAM, set DLENGTH to the top memory address first.

Every microinstruction period, the Address Generator will decrement DBASE and output a new address. It will put all zeros on its RAM data bus, and initiate a write cycle. When DBASE has counted down to zero, the RAM Clear bit will clear automatically to indicate to the Host that memory has been cleared.

The action taken by the ESP when RAM clearing is finished depends on the state of the Refresh Disable bit. If the bit is false, then the ESP will begin refreshing RAM when

the clearing operation is complete. This is the normal course of action, since the RAM which has been cleared must be refreshed to preserve its cleared state.

Host RAM Initialization

When the ESP is Halted, the Host processor has free access to the ESP's RAM through its Host RAM Interface. The exact protocol for accessing RAM through these registers is detailed in section 5.1.3.

Access to RAM is available through the Host Interface RAM address and data registers only when the ESP is Halted. Host table preload write operations will always have priority over refreshing operations.

5.4.2 Power-up Sequence

1. Bring HALT high. Write $FF to the Halt Enable Register to halt the ESP.
2. Set MEMSIZ to match system's physical RAM configuration. ($0000FF for 64K)
3. Set DLENGTH register to all of memory. ($FFFFFF)
4. Reset the RAM Clear bit to 0.
5. Set the RAM Clear bit to 1 and the Refresh Disable bit to 0 in the Host Signal Control Register. (Write a value of 2).
6. Wait for the RAM clear bit to reset.
7. Initialize Host Serial Control Register.
8. Set the Product Shift Mode to 1 by setting high all bits in the SIGREG SPR.
9. Clear all serial I/O registers.

5.4.3 Program Load Sequence

1. Load and/or initialize GPRs.
2. Load microcode.
3. Load tables (if any).
4. Park host GPR access register at $FF.

PART 6 ... ESP PIN DESCRIPTION

6.1 Pinout

RAM Interface (total 21)

Pin Name	Function	# Pins
RAM[15:0]	RAM Address/Data	16
DWEN	RAM or I/O Write Enable (active low)	1
RAS	Row Address Strobe	1
CAS	Column Address Strobe	1
GATE	Address Gate Enable	1
I/OSEL	I/O Select (active low)	1

Host Interface (total 12)

Pin Name	Function	# Pins
HOST[7:0]	Host Address/Data	8
CS	Chip Select (active low)	1
R/W	Host Read/Write	1
AS	Host Address strobe	1
HALT	ESP halt control	1

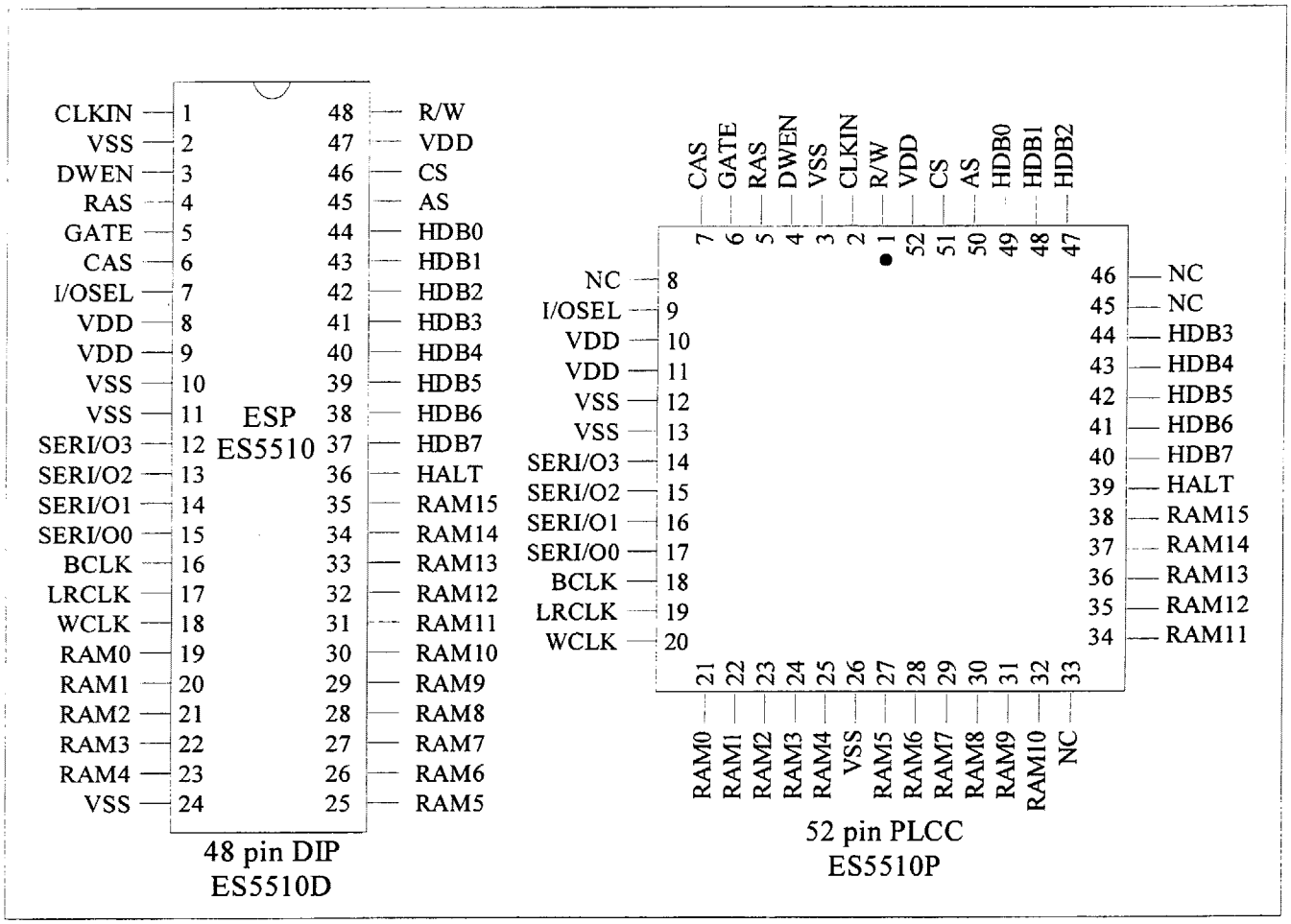
Serial Interface (total 7)

Pin Name	Function	# Pins
SER[3:0]	Serial Data I/O	4
BCLK	Bit Clock	1
WCLK	Word Clock	1
LRCLK	Left/Right Clock	1

Other (total 8)

Pin Name	Function	# Pins
CLK_IN	System Clock	1
VDD	+5 volt power supply	3
VSS	Ground (0 volt)	4

ESP Pinouts



6.2 Pin Descriptions

6.2.1 Host Interface

R/W — Read/Write input. Driven by the host microprocessor to a high level when reading ESP registers, and to a low level when writing them.

/CS — Chip Select input. Driven low by host to access ESP registers for read or write.

/AS — Address Strobe input. The host data bus is used to transfer both address and data. A high to low transition on AS/ is required for the ESP to latch the address from the host.

HALT — Halt input. Halt stops ESP microprogram execution at the next completion of the microcode, and allows ultimate host access to ESP registers. On the falling edge, Halt restarts the ESP microprogram execution. HALT can be used to synchronize microprogram execution to external events.

HOST[7:0]	Host address inputs/bidirectional data I/O. Used for reading and writing ESP registers.

6.2.2 External RAM

RAS	RAM Row Address Strobe output. Row addresses from the ESP are identified by high-to-low transitions on this pin. RAS occurs every instruction cycle.
GATE	Gate output. The falling edge of GATE is used to latch ESP column addresses. This is necessary because data and not column addresses are being transferred at the normal CAS time. GATE occurs every instruction cycle.
CAS	RAM Column Address Strobe output. During an external RAM read, CAS low indicates when to enable read data onto the ESP RAM data bus. During an external RAM write operation, the negative edge of CAS is delayed and is signals when RAM write data is valid.
DWEN	External RAM Write Enable output. When low, indicates an external RAM or I/O write operation will occur instead of the default read.
IO_SEL	External I/O Select output. This active low signal selects external I/O space rather than RAM. IO_SEL replaces CAS during I/O accesses.
RAM[15:0]	Multiplexed bidirection external RAM and I/O address/data. This bus contains row and column address information multiplexed to coincide with RAS, GATE, and CAS or IO_SEL timing. The direction of data is determined by the DWEN line. RAM or I/O accesses occur once every instruction cycle.

6.2.3 Serial Interface

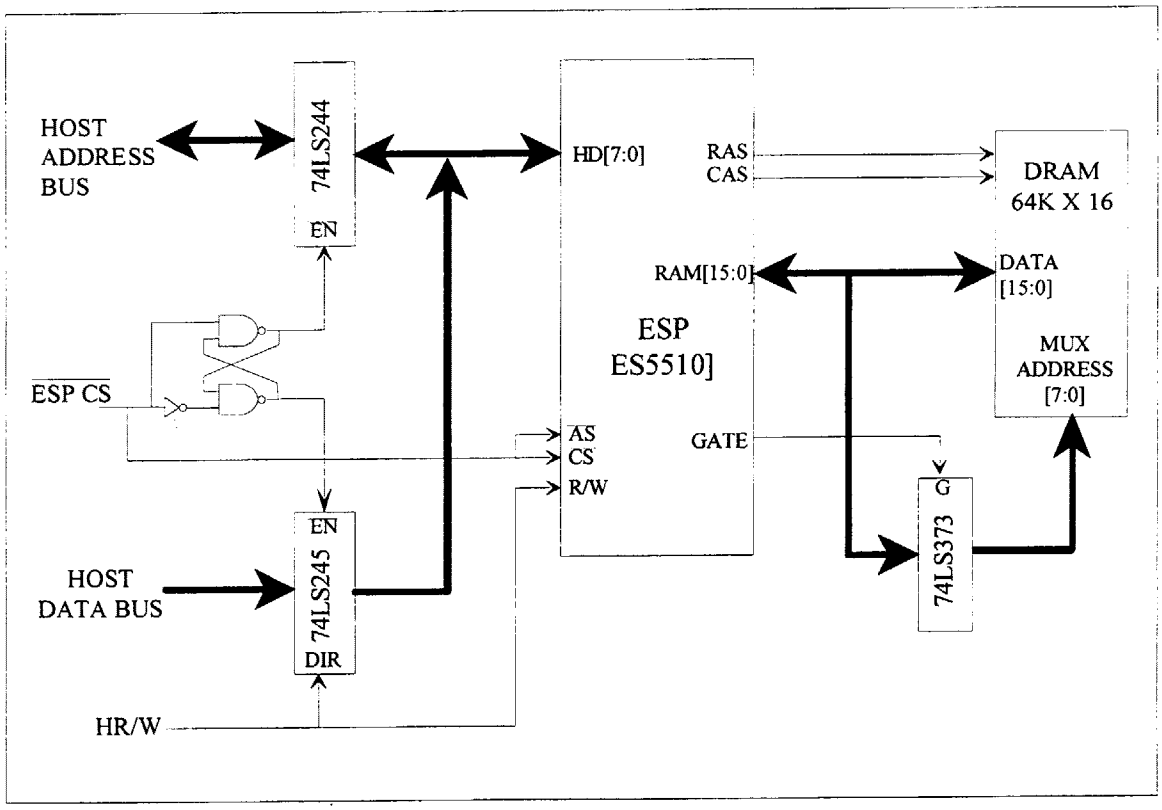
The signals BCLK, WCLK, and LRCLK can be either outputs or inputs, depending on whether the ESP is programmed to operate as the serial master or as a slave.

BCLK	Bit Clock. This is the bit rate clock for all serial data interface lines.
WCLK	Word Clock. A negative transition on WCLK signals the end of a 16 bit data transfer on the serial data pins. See section 5.1.7 for additional information. WCLK is generally not used in the I2S serial mode.
LRCLK	Left/Right Clock. This pin is used to initiate data transfer on the serial data lines and to steer data to the proper (left or right) register on serial data inputs. A positive transition of LRCLK initiates a left word transfer, while a negative transition initiates a right word transfer.
SER[0:3]	Bidirectional Serial Data Lines. Serial data is transferred on these pins. The pins are independently programmable as inputs or outputs.

6.2.4 Supplies (clock and power)

CLK_IN	Primary Clock input.
VDD	+5.0 volt power supply < 100mA @ 10MHz clock
VSS	Ground. (0 volt)

6.2.5 System Interface Hardware Block Diagram



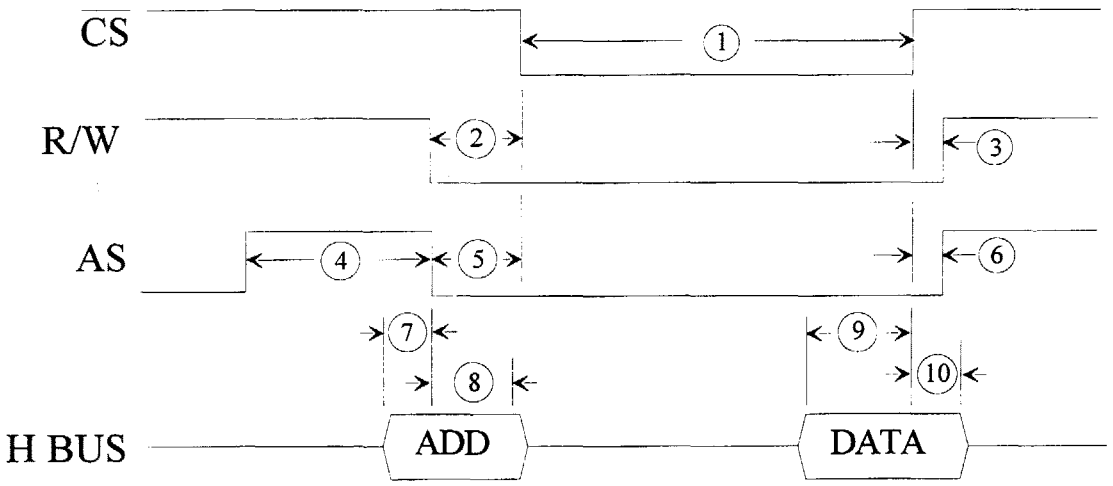
PART 7 ... Timing Diagram

ESPR7 TIMING SPECIFICATIONS

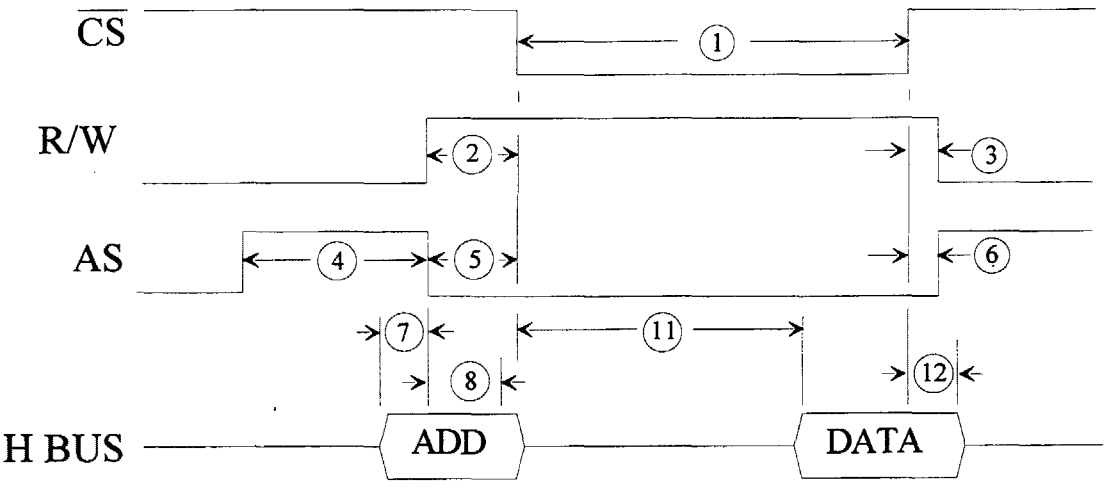
			8Mhz			10Mhz			12Mhz		
#	Characteristic	Symbol	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
1	Chip Select Active Low	CSL	100		1000	100		1000	100		1000
2	Read/Write setup time	R/WSU	20			20			20		
3	Read/Write hold after CS	R/WH	20			20			20		
4	Address Strobe High Time	ASHT	50			50			50		
5	Address Strobe to CS setup time	ASSU	30			30			30		
6	Address Strobe hold time	ASH	0			0			0		
7	Address In setup time	ADSU	30			30			30		
8	Address In hold time	ADH	30			30			30		
9	Data In Set up time to CS rise	DSU	30			30			30		
10	Data In hold time	DH	30			30			30		
11	Data Out Access time	DACC			75			75			75
12	Memory Cycle time	TCYC		375			300			250	
13	RAS high time	RASH	120	125	130	95	100	105	80	83	86
14	RAS to GATE delay	RGD	60	63	65	47	50	53	40	42	45
15	GATE high	GATE	120	125	130	95	100	105		83	
16	Memory R/W to RAS setup	MRWS	60	63	65	47	50	53	40	42	45
17	CAS high time WRITE	CASHW	180	186	192	145	150	155	121	125	130
18	CAS high time READ	CASHR		250			200			165	
19	Row Address setup time	RASU		40			35			30	
20	Row Address hold time	RAH		30			25			20	
21	Column Address setup time	CASU		25			20			15	
22	Column Address hold time	CAH		15			15			15	
23	Memory Read Data setup time	MRDSU	30			30			30		
24	Memory Read Data hold time	MRDH	10			10			10		
25	Memory Write Data setup time	MWDSU		50			40			30	
26	Memory Write Data hold time	MWDH		20			15			10	
27	Bclk rising output delay	BCLKRD		20	30		20	30		20	30
28	Bclk falling output delay	BCLKFD		20	30		20	30		20	30
29	LRclk and Wclk delay	LRWD		20	30		20	30		20	30
30	Bclk to Serial Data out delay	SDD		30	40		30	40		30	40
31	Serial Data Out hold time	SDOH	20			20			20		
32	Data In setup time	SDSU	30			30			25		
33	Data In hold time	SDIH	30			30			30		
34	LRclk & Wclk after Bclk ↓	LRWC	10		30	10		30	10		30
35	LRclk & Wclk before Bclk ↑	LRWSU	50			50			50		
36	Data In setup to Bclk ↓ (Sony)	SSDISU	40			40			40		
37	Data In hold after Bclk ↓ (Sony)	SSDIH	40			40			40		

Note: All timing parameters are in nanoseconds.

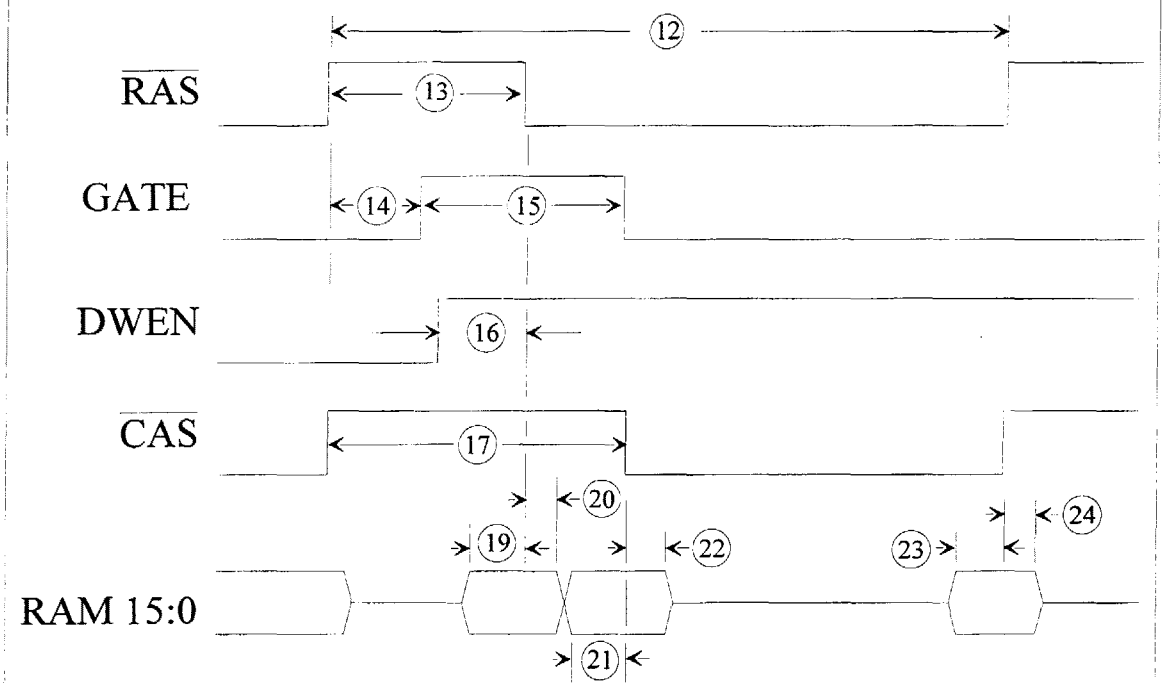
CPU WRITE to ESP TIMING



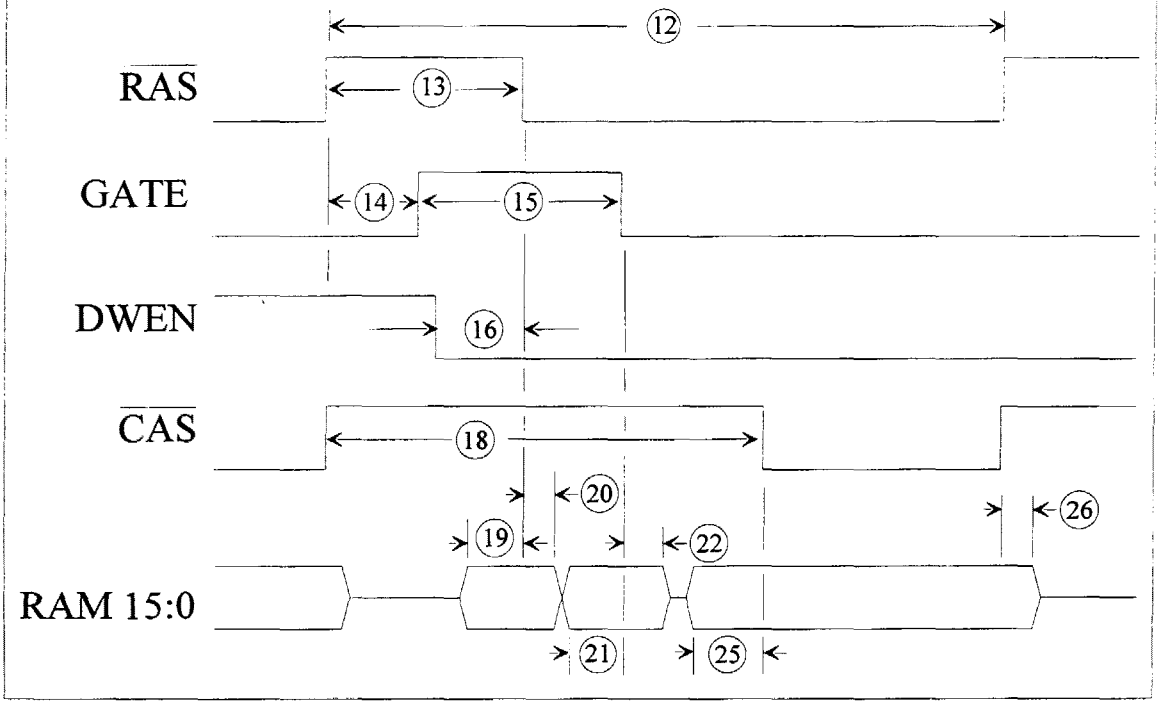
CPU READ from ESP TIMING

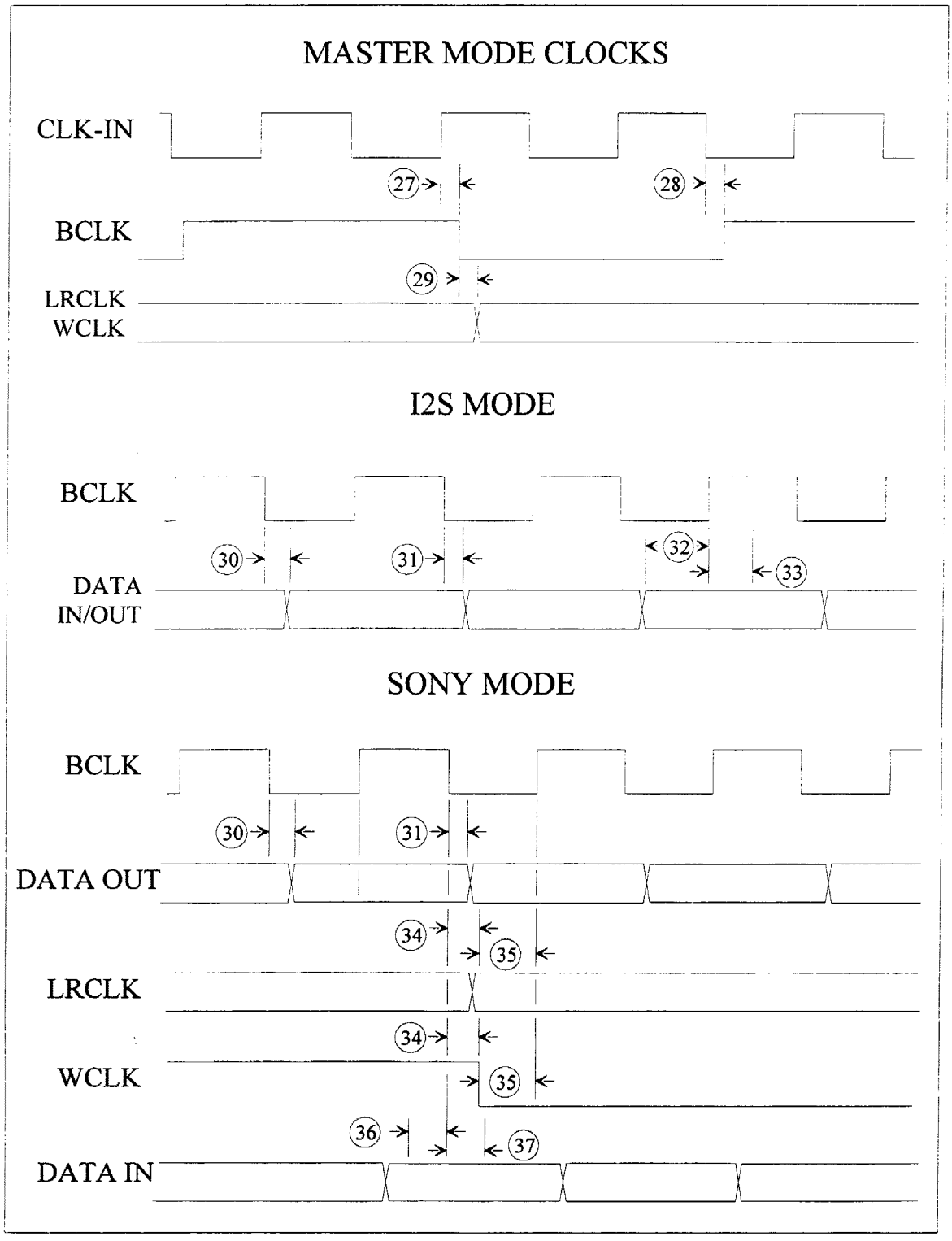


DRAM READ INTERFACE TIMING

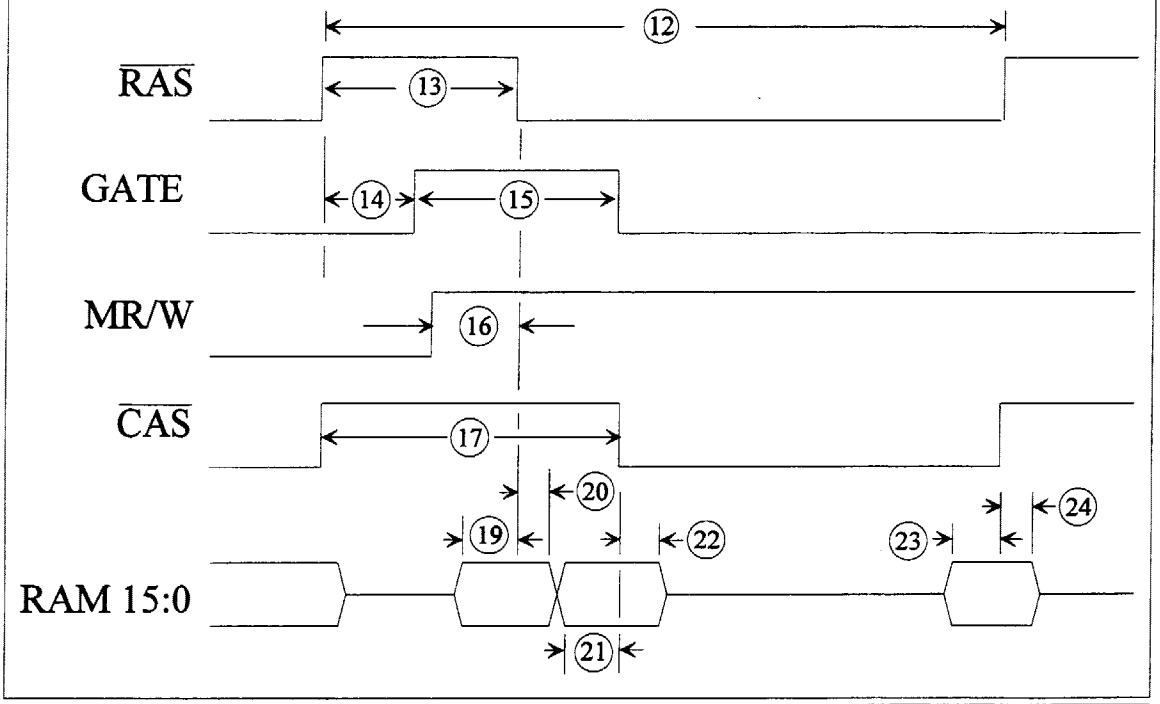


DRAM WRITE INTERFACE TIMING

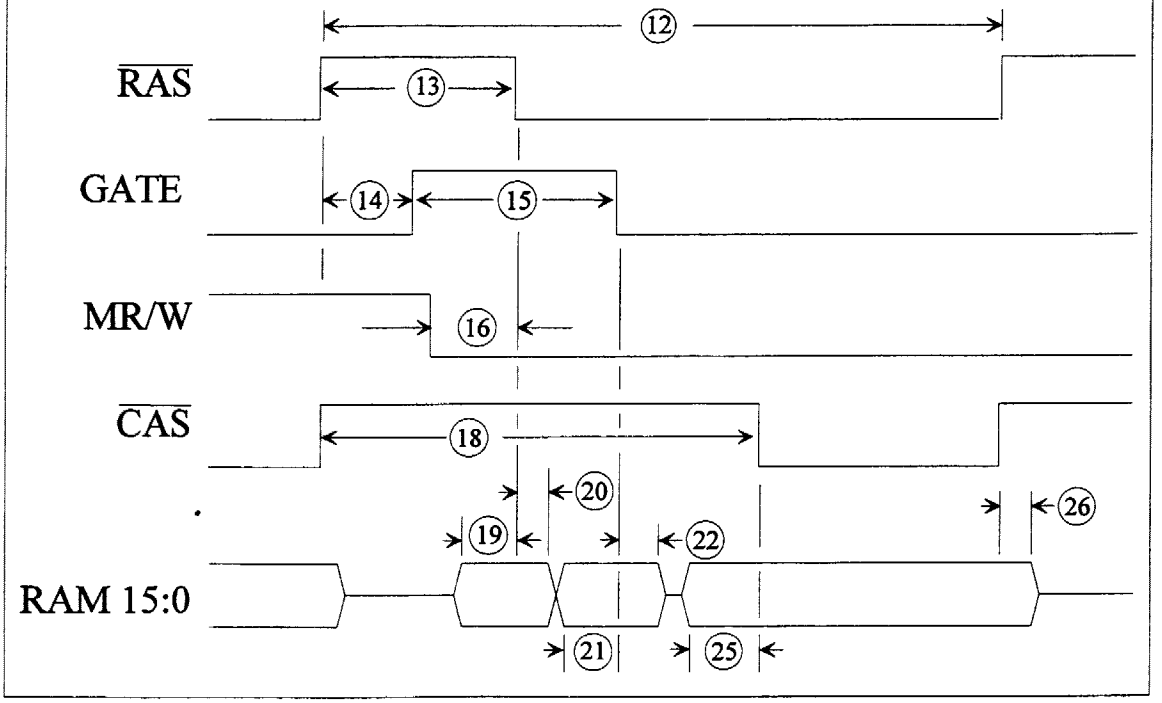


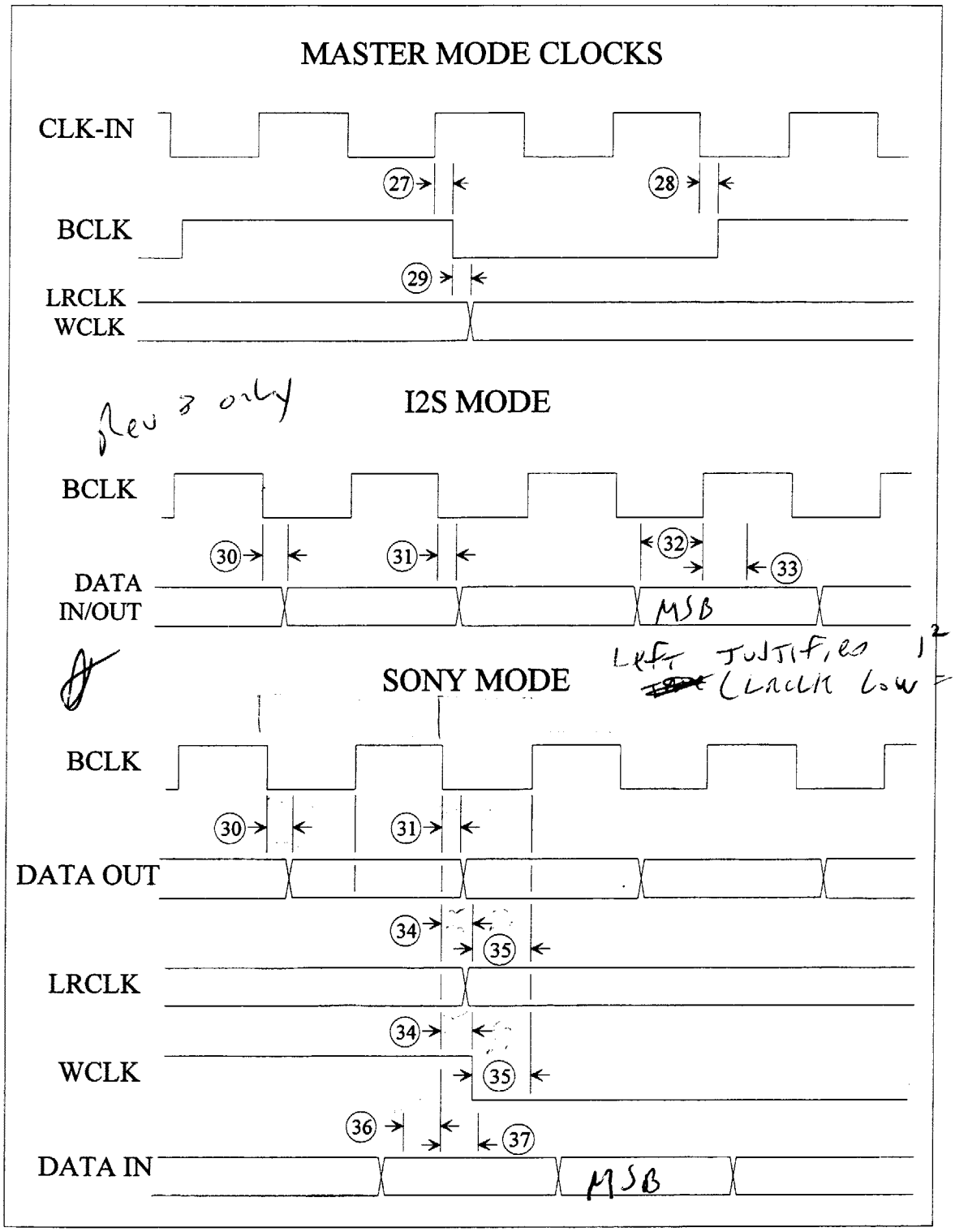

MASTER MODE CLOCKS
CLK-IN
27
28
BCLK
29
LRCLK
WCLK
I2S MODE
BCLK
30
31
32
33
DATA
IN/OUT
SONY MODE
BCLK
30
31
DATA OUT
34
35
LRCLK
34
WCLK
35
36
37
DATA IN

DRAM READ INTERFACE TIMING



DRAM WRITE INTERFACE TIMING




MASTER MODE CLOCKS
CLK-IN
27
28
BCLK
29
LRCLK
WCLK
Rev B only
I2S MODE
BCLK
30
31
32
33
DATA
IN/OUT
MSB
LEFT JUSTIFIED I2S
(LRCLK Low = RIGHT)
SONY MODE
BCLK
30
31
DATA OUT
34
35
LRCLK
34
WCLK
35
36
37
DATA IN
MSB