pdp11

KD11-E central processor maintenance manual

digital

EK-KD11E-TM-001

KD11-E central processor maintenance manual

digital equipment corporation • maynard, massachusetts

1st Edition, December 1976

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

ILLUSTRATIONS

ILLUSTRATIONS (Cont)

TABLES

PREFACE

This manual describes the KD11-E Central Processing Unit (M7265 and M7266). The user must have a general knowledge of digital circuitry and a basic understanding of PDP-11 computers to completely understand the contents of this manual.

The following related documents may be valuable as references:

PDP-11 Peripherals Handbook
PDP-11/34 Processor Handbook
PDP-11/34 System User's Guide (EK-11034-OP-001)
KD11-E Print Set (MP00043)

CHAPTER 1
OVERALL DESCRIPTION

The KD11-E is a 2-board central processing unit (CPU) that is combined with a memory system, Unibus terminators, and optional peripherals in a DD11-P backpanel to build a basic PDP-11/34 computer. The unit connects directly to the Unibus as a subsystem, and is capable of controlling the time allocation of the Unibus for peripherals, performing arithmetic and logic operations, and decoding instructions. It can perform data transfers directly between I/O devices and memory, do both single- and double-operand addressing, handle both 16-bit word and 8-bit byte data, and address up to 128K of Unibus address space via a memory management system.

The KD11-E is program-compatible with both the KD11-A (PDP-11/35 and PDP-11/40 computer systems) and the LSI-11 (with the inclusion of the two special LSI-11 instructions). It contains the KT11-D Memory Management System (optional with the KD11-A, not offered with the LSI-11) and executes the Extended Instruction Set (EIS) instructions, which were optional with the KD11-A and standard with the LSI-11. The KD11-E does not execute the Floating Instruction Set (FIS).

CHAPTER 2
INSTRUCTION SET

2.1 INTRODUCTION

The KD11-E is defined by its instruction set. The sequences of processor operations are selected according to the instruction decoding. The following describes the PDP-11/34 instructions and instruction set addressing modes along with instruction set differences from those of the KD11-A, KD11-B, and KD11-D.

2.2 ADDRESSING MODES

Data stored in memory must be accessed and manipulated. Data handling is specified by a PDP-11/34 instruction (MOV, ADD, etc.), which usually indicates:

1. The function (operation code)

2. A general-purpose register to be used when locating the source operand and/or locating the destination operand

3. An addressing mode (to specify how the selected register(s) is to be used)

Because a large portion of the data handled by a computer is usually structured (in character strings, in arrays, in lists, etc.), the PDP-11/34 has been designed to handle structured data efficiently and flexibly. The general registers may be used with an instruction in any of the following ways:

1. As accumulators. The data to be manipulated resides within the register.

2. As pointers. The contents of the register are the address of the operand, rather than the operand itself.

3. As pointers, which automatically step through core locations. Automatically stepping forward through consecutive core locations is known as autoincrement addressing; automatically stepping backward is known as autodecrement addressing. These modes are particularly useful for processing tabular data.

4. As index registers. In this instance the contents of the register and the word following the instruction are summed to produce the address of the operand. This allows easy access to variable entries in a list.

PDP-11/34s also have instruction addressing mode combinations that facilitate temporary data storage structures for convenient handling of data which must be frequently accessed. This is known as the "stack."

In the PDP-11/34, any register can be used as a "stack pointer" under program control; however, certain instructions associated with subroutine linkage and interrupt service automatically use Register 6 as a "hardware stack pointer." For this reason, R6 is frequently referred to as the "SP."

R7 is used by the processor as its program counter (PC).

Two types of instructions utilize the addressing modes: single-operand and double-operand. Figure 2-1 shows the formats of these two types of instructions. The addressing modes are listed in Table 2-1.

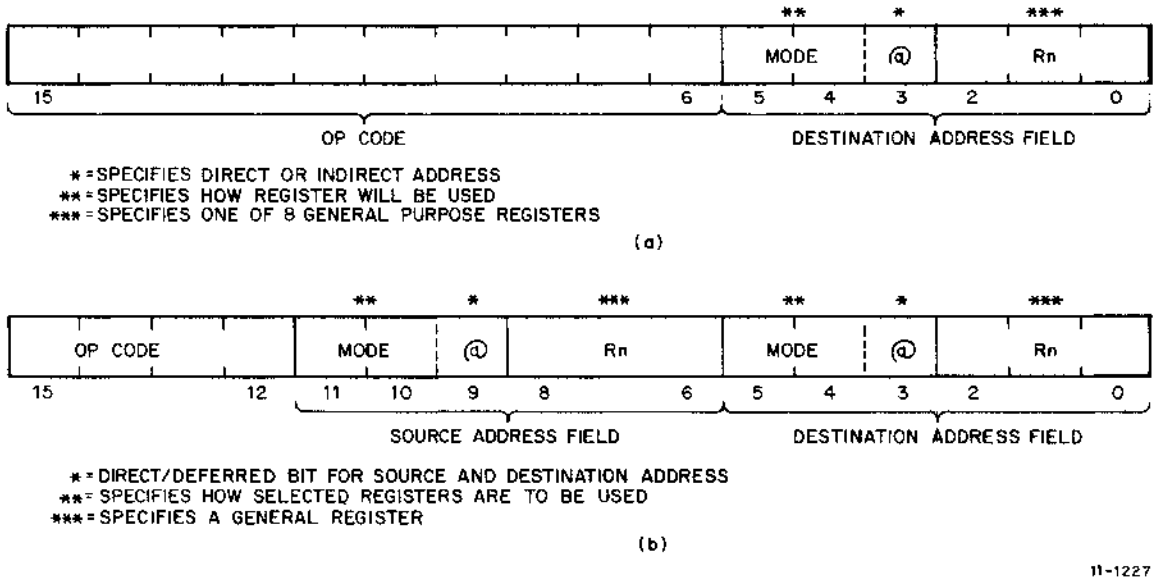


Figure 2-1 Addressing Mode Instruction Formats

Table 2-1 Addressing Modes

Mode	Binary Code	Name	Assembler Syntax*	Function
			Direct Modes	
0	000	Register	Rn	Register contains operand.
2	010	Autoincrement	(Rn)+	Register contains address of operand. Register contents incremented after reference.
4	100	Autodecrement	–(Rn)	Register contents decremented before reference register contains address of operand.
6	110	Index	X(Rn)	Value X (stored in a word following the instruction) is added to (Rn) to produce address of operand. Neither X nor (Rn) is modified.
			Deferred Modes	
1	001	Register Deferred	@Rn or (Rn)	Register contains the address of the operand.
3	011	Autoincrement Deferred	@(Rn)+	Register is first used as a pointer to a word containing the address of the operand, then incremented (always by two, even for byte instructions).
5	101	Autodecrement Deferred	@–(Rn)	Register is decremented (always by two, even for byte instructions) and then used as a pointer to a word containing the address of the operand.
7	111	Index Deferred	@X(Rn)	Value X (stored in the memory word following the instruction) and (Rn) are added and the sum is used as a pointer to a word containing the address of the operand. Neither X nor (Rn) is modified.

Table 2-1 Addressing Modes (cont)

Mode	Binary Code	Name	Assembler Syntax*	Function
			PC Addressing	
2	010	Immediate	#n	Operand follows instruction.
3	011	Absolute	@#A	Absolute address follows instruction.
6	110	Relative	A	Address of A, relative to the instruction, follows the instruction.
7	111	Relative Deferred	@A	Address of location containing address of A, relative to the instruction, follows the instruction.

* Rn = Register
X, n, A = next program counter (PC) word (constant)

2.3 PDP-11/34 INSTRUCTIONS

The PDP-11/34 instructions can be divided into five groups:

1. Single-Operand Instructions (shifts, multiple precision instructions, rotations)
2. Double-Operand Instructions (arithmetic and logical instructions)
3. Program Control Instructions (branches, subroutines, traps)
4. Operate Group Instructions (processor control operations)
5. Condition Code Operators (processor status word bit instructions)

Tables 2-2 through 2-6 list each instruction, including byte instructions for the respective instruction groups. Figure 2-2 shows the six different instruction formats of the instruction set, and the individual instructions in each format.

Table 2-2 Single Operand Instructions

Mnemonic	OP Code	Operation	Condition Codes	Description
CLR CLRB Clear	0050DD* 1050DD	(dst)† ← 0	N: cleared Z: set V: cleared C: cleared	Contents of specified destination are replaced with zeroes.
COM COMB Complement	0051DD 1051DD	(dst) ← n (dst)	N: set if most significant bit of result is 0 Z: set if result is 0 V: cleared C: set	Replaces the contents of the destination address by their logical complement (each bit equal to 0 set and each bit equal to 1 cleared).
INC INCB Increment	0052DD 1052DD	(dst) ← (dst) + 1	N: set if result is less than 0 Z: set if result is 0 V: set if (dst) was 077777 C: not affected	Add 1 to the contents of the destination.
DEC DECB Decrement	0053DD 1053DD	(dst) ← (dst) - 1	N: set if result is less than 0 Z: set if result is 0 V: set if (dst) was 100000 C: not affected	Subtract 1 from the contents of the destination.
NEG NEGB Negate	0054DD 1054DD	(dst) ← - (dst)	N: set if result is less than 0 Z: set if result is 0 V: set if result is 100000 C: cleared if result is 0	Replaces the contents of the destination address by its 2's complement. Note that 100000 is replaced by itself.
ADC ADCB Add Carry	0055DD 1055DD	(dst) ← (dst) + C	N: set if result is less than 0 Z: set if result is 0 V: set if (dst) is 077777 and C is 1 C: set if (dst) is 177777 and C is 1	Adds the contents of the C-bit into the destination. This permits the carry from the addition of the low-order words/bytes to be carried into the high-order results.

Table 2-2 Single Operand Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
SBC SBCB Subtract Carry	0056DD 1056DD	(dst) ← (dst) -C	N: set if result is less than 0 Z: set if result is 0 V: set if (dst) was 100000 C: cleared if (dst) is 0 and C is 1	Subtracts the contents of the C-bit from the destination. This permits the carry from the subtraction of the low order words/bytes to be subtracted from the high-order part of the result.
TST TSTB Test	0057DD 1057DD	(dst) ← (dst)	N: set if result is less than 0 Z: set if result is 0 V: cleared C: cleared	Sets the condition codes N and Z according to the contents of the destination address.
ROR RORB Rotate Right	0060DD	(dst) ← (dst) rotate right one place.	N: set if high-order bit of the result is set Z: set if all bits of result are 0 V: loaded with the exclusive-OR of the N-bit and the C-bit as set by ROR	Rotates all bits of the destination right one place. The low-order bit is loaded into the C-bit and the previous contents of the C-bit are loaded into the high-order bit of the destination.
ROL ROLB Rotate Left	0061DD 1061DD	(dst) ← (dst) rotate left one place.	N: set if the high order bit of the result word is set (result < 0); cleared otherwise Z: set if all bits of the result word = 0; cleared otherwise V: loaded with the exclusive-OR of the N-bit and C-bit (as set by the completion of the rotate operation) C: loaded with the high order bit of the destination	Rotate all bits of the destination left one place. The high-order bit is loaded into the C-bit of the status word and the previous contents of the C-bit are loaded into the low-order bit of the destination.

Table 2-2 Single Operand Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
ASR ASRB Arithmetic Shift Right	0062DD 1062DD	(dst) ← (dst) shifted one place to the right.	N: set if the high order bit of the result is set (result < 0); cleared otherwise Z: set if the result = 0; cleared otherwise V: loaded from the exclusive-OR of the N-bit and C-bit (as set by the completion of the shift operation). C: loaded from low order bit of the destination	Shifts all bits of the destination right one place. The high-order bit is replicated. The C-bit is loaded from the low-order bit of the destination. ASR performs signed division of the destination by two.
ASL ASLB Arithmetic Shift Left	0063DD 1063DD	(dst) ← (dst) shifted one place to the left.	N: set if high-order bit of the (result < 0); cleared otherwise Z: set if the result = 0; cleared otherwise V: loaded with the exclusive-OR of the N-bit and C-bit and C-bit (as set by the completion of the shift operation) C: loaded with the high-order bit of the destination	Shifts all bits of the destination left one place. The low-order bit is loaded with a 0. The C-bit of the status word is loaded from the high-order bit of the destination. ASL performs a signed multiplication of the destination by 2 with overflow indication.

Table 2-2 Single Operand Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
ASH Arithmetic Shift	072RSS	R ← R Shifted Arithmetically NN places to right or left Where NN = (src)	N: set if result <0; cleared otherwise. Z: set if result = 0; cleared otherwise. V: set if sign of register changed during shift; cleared otherwise. C: loaded from last bit shift out of register.	The contents of the register are shifted right or left the number of times specified by the source operand. The shift count is taken as the low-order 6 bits of the source operand. This number ranges from -32 to +31. Negative is a right shift and positive is a left shift. See Paragraph 2.5 for example.
ASHC Arithmetic Shift Combined	073RSS	R, Rvl ← R, Rvl The double word is shifted NN places to the right or left, where NN = (src)	N: set if result <0; cleared otherwise. Z: set if result = 0; cleared otherwise. V: set if sign bit changes during the shift; cleared otherwise. C: loaded with high-order bit when right shift (loaded with the last bit shifted out of the 32-bit operand).	The contents of the register and the register ORed with one are treated as one 32-bit word. Rvl (bits 0–15) and R (bits 16–31) are shifted right or left the number of times specified by the shift count. The shift count is taken as the low-order 6 bits of the source operand. This number ranges from -32 to +31. Negative is a right shift and positive is a left shift. When the register chosen is an odd number, the register and the register ORed with one are the same. In this case, the right shift becomes a rotate. The 16-bit word is rotated right the number of bits specified by the shift count. See Paragraph 2.5 for example.
SXT Sign Extend	0067DD	(dst) ← 0 if N bit is clear (dst) ← -1 N bit is set	N: unaffected Z: set if N bit clear V: cleared C: unaffected	If the condition code bit N is set then a -1 is placed in the destination operand: if N bit is clear, then a 0 is placed in the destination operand. This instruction is particularly useful in multiple precision arithmetic because it permits the sign to be extended through multiple words.

Table 2-2 Single Operand Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
SWAB Swap Byte	0003DD	Byte 1/Byte 0 Byte 0/Byte 1	N: set if high-order bit of low-order byte (bit 7) of result is set; cleared otherwise. Z: set if low-order byte of result = 0; cleared otherwise. V: cleared C: cleared	Exchanges high-order byte and low-order byte of the destination word (destination must be a word address).

Table 2-3 Double Operand Instructions

Mnemonic	OP Code	Operation	Condition Codes	Description
MOV MOVB Move	01SSDD* 11SSDD	(dst) ← (src) †	N: set if (src) < 0, cleared otherwise Z: set if (src) = 0; cleared otherwise V: cleared C: not affected	Word: Moves the source operand to the destination location. The previous contents of the destination are lost. The source operand is not affected. Byte: Same as MOV. The MOVB to a resistor (unique among byte instructions) extends the most significant bit of the low order byte (sign extension). Otherwise MOVB operates on bytes exactly as MOV operates on words.
CMP DMPB Compare	02SSDD 12SSDD	(src) - (dst) [in detail, (src) + ~ (dst) + 1]	N: set if result < 0, cleared otherwise Z: set if result = 0; cleared otherwise V: set if there was arithmetic overflow (i.e., operands were of opposite signs and the sign of the destination was the same as the sign of the result); cleared otherwise. C: cleared if there was a carry from the most significant bit of the result; set otherwise	Compares the source and destination operands and sets the condition codes which may then be used for arithmetic and logical conditional branches. Both operands are unaffected. The only action is to set the condition codes. The compare is customarily followed by a conditional branch instruction. Note that unlike the subtract instruction the order of operation is (src) - (dst), not (dst) - (src).

Table 2-3 Double Operand Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
BIT BITB Bit Test	03SSDD 13SSDD	(src) ⋀ (dst)	N: set if high order bit of result set; cleared otherwise Z: set if result = 0; cleared otherwise V: cleared C: not affected	Performs logical AND comparison of the source and destination operands and modifies condition codes accordingly. Neither the source nor destination operands are affected. The BIT instruction may be used to test whether any of the corresponding bits that are set in the destination are clear in the source.
BIC BICB Bit Clear	04SSDD 14SSDD	(dst) ← ~ (src) ⋀ (dst)	N: set if high order bit of result set; cleared otherwise Z: set if result = 0; cleared otherwise V: cleared C: not affected	Clears each bit in the destination that corresponds to a set bit in the source. The original contents of the destination are lost. The contents of the source are unaffected.
BIS BISB Bit Set	05SSDD 15SSDD	(dst) ← (src) ⋀ (dst)	N: set if high order bit of result set; cleared otherwise Z: set if result = 0; cleared otherwise V: cleared C: not affected	Performs inclusive-OR operation between the source and destination operands and leaves the result at the destination address; i.e., corresponding bits set in the destination. The contents of the destination are lost.
ADD Add	06SSDD	(dst) ← (src) + (dst)	N: set if result 0; cleared otherwise Z: set if result = 0; cleared otherwise	Adds the source operand to the destination operand and stores the result at the destination address. The original contents of the destination are lost. The contents of the source are not affected. Two's complement addition is performed.

Table 2-3 Double Operand Instruction (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
ADD (Cont)			V: set if there was arithmetic overflow as a result of the operation (that is, both operands were of the same sign and the result was of the opposite sign); cleared otherwise. C: set if there was a carry from the most significant bit of the result; cleared otherwise.	
SUB Subtract	16SSDD	(dst) ← (dst) - (src) in detail, (dst) + ~ (src) + 1 (dst)	N: set if result < 0; cleared otherwise Z: set if result = 0; cleared otherwise V: set if there was arithmetic overflow as a result of the operation (i.e., if operands were of opposite signs and the sign of the source was the same as the sign of the result); cleared otherwise C: cleared if there was a carry from the most significant bit of the result; set otherwise	Subtracts the source operand from the destination operand and leaves the result at the destination address. The original contents of the destination are lost. The contents of the source are not affected. In double precision arithmetic, the C-bit, when set, indicates a borrow.

* SS = source (address mode and register)

† (src) = source contents

Table 2-3 Double Operand Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
MUL Multiply	070RSS	R, Rvl ← Rx(src)	N: set if product is <0; cleared otherwise. Z: set if product is 0; cleared otherwise. V: cleared C: set if the result is less than -2^{15} or greater than or equal to $2^{15}-1$.	The contents of the destination register and source taken as two's complement integers are multiplied and stored in the destination register and the succeeding register (if R is even). If R is odd, only the low-order product is stored. Assembler syntax is: MUL S,R. (Note that the actual destination is R, Rvl which reduces to just R when R is odd.) (See Paragraph 2.5.1 for example).
DIV Divide	071RSS	R, Rvl ← R, Rvl (src)	N: set if quotient <0; cleared otherwise. Z: set if quotient = 0; cleared otherwise. V: set if source = 0 or if the absolute value of the register is larger than the absolute value of the source. (In this case the instruction is aborted because the quotient would exceed 15 bits.) C: set if divide 0 attempted; cleared otherwise.	The 32-bit two's complement integer in R and Rvl is divided by the source operand. The quotient is left in R; the remainder is of the same sign as the dividend. R must be even. (See Paragraph 2.5.2 for example.)
XOR	074RDD	(dst) ← Rv (dst)	N: set if the result <0; cleared otherwise. Z: set if result = 0; cleared otherwise. V: cleared C: unaffected	The exclusive OR of the register and destination operand is stored in the destination address. Contents of register are unaffected. Assembler format is XOR R,D.

Table 2-4 Program Control Instructions

Mnemonic	OP Code	Operation	Condition Codes	Description
BR Branch	000400 xxx†	PC ← PC + (2 X offset)	Unaffected	Provides a way of transferring program control within a range of - 128 to +127 words with a one word instruction. It is an unconditional branch.
BNE Branch if not equal	001000 xxx	PC ← PC + (2 X offset) if Z = 0	Unaffected	Tests the state of the Z-bit and causes a branch if the Z-bit is is clear. BNE is the complementary operation to BEQ. It is used to test inequality following a CMP, to test that some bits set in the destination were also in the source, following a BIT, and generally, to test that the result of the previous operation was not 0.
BEQ Branch if equal	001400 xxx	PC ← PC + (2 X offset) if Z = 1	Unaffected	Tests the state of the Z-bit and causes a branch if Z is set. As an example, it is used to test equality following a CMP operation. to test that no bits set in the destination were also set in the source following a BIT operation, and generally, to test that the result of the previous operation was 0.
BGE Branch if greater than or equal	002000 xxx	PC ← PC + (2 X offset) if N v V = 0	Unaffected	Causes a branch if N and V are either both clear or both set. BGE is the complementary operation to BLT. Thus. BGE always causes a branch when it follows an operation that caused addition to two positive numbers. BGE also causes a branch on a 0 result.

Table 2-4 Program Control Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
BLT Branch if less than	002400 xxx	PC ← PC + (2 X offset) if N V = 1	Unaffected	Causes a branch if the exclusive-OR of the N- and V-bits are 1. Thus. BLT always branches following an operation that added two negative numbers, even if overflow occurred. In particular, BLT always causes a branch if it follows a CMP instruction operating on a negative source and a positive destination (even if overflow occurred). Further, BLT never causes a branch when it follows a CMP instruction operating on a positive source and negative destination. BLT does not cause a branch if the result of the previous operation was 0 (without overflow).
BGT Branch if greater than	003000 xxx	PC ← PC + (2 X offset) if Z v (N⊻ V) = 0	Unaffected	Operation of BGT is similar to BGE, except BGT does not cause a branch on a 0 result.
BLE Branch if less than or equal to	003400 xxx	PC ← PC + (2 X offset) if Z v (N⊻V) = 1	Unaffected	Operation is similar to BLT, but in addition will cause a branch if the result of the previous operation was 0.
BPL Branch if plus	100000 xxx	PC ← PC + (2 X offset) if N = 0	Unaffected	Tests the state of the N-bit and causes a branch if N is clear. BPL is the complementary operation of BMI.
BMI Branch if minus	100400 xxx	PC ← PC + (2 X offset) if N = 1	Unaffected	Tests the state of the N-bit and causes a branch if N is set. It is used to test the sign (most significant bit) of the result of the previous operation.

Table 2-4 Program Control Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
BHI Branch if higher	101000 xxx	PC ← PC + (2 X offset) if C = 0	Unaffected	Causes a branch if the previous operation causes neither a carry nor a 0 result. This will happen in comparison (CMP) operations as long as the source has a higher unsigned value than the destination.
BLOS Branch if lower or same	101400 xxx	PC ← PC + (2 X offset) if C v Z = 1	Unaffected	Causes a branch if the previous operation caused either a carry or a 0 result. BLOS is the complementary operation to BHI. The branch occurs in comparison operations as long as the source is equal to or has a lower unsigned value than the destination. Comparison of unsigned values with the CMP instruction to be tested for "higher or same" and "higher" by a simple test of the C-bit.
BVC Branch if V-bit clear	102000 xxx	PC ← PC + (2 X offset) if V = 0	Unaffected	Tests the state of the V-bit and causes a branch if the V-bit is clear. BVC is complementary operation to BVS.
BVS Branch if V-bit set	102400 xxx	PC ← PC + (2 X offset) if V = 1	Unaffected	Tests the state of V-bit (overflow) and causes a branch if the V-bit is set. BVS is used to detect arithmetic overflow in the previous operation.
BCC } BHIS } Branch if carry clear Branch if higher than the same	103000 xxx	PC ← PC + (2 X offset) if C = 0	Unaffected	Tests the state of the C-bit and causes a branch if C is clear. BCC is the complementary operation to BCS.
BCS } BLO } Branch if carry set Branch if lower	103400 xxx	PC ← PC + (2 X offset) if C = 1	Unaffected	Tests the state of the C-bit and causes a branch if C is set. It is used to test for a carry in the result of a previous operation.

Table 2-4 Program Control Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
JMP Jump	0001DD	PC ← (dst)	Unaffected	JMP provides more flexible program branching than provided with the branch instruction. Control may be transferred to any location in memory (no range limitation) and can be accomplished with the full flexibility of the addressing modes, with the exception of register mode 0. Execution of a jump with mode 0 will cause an illegal instruction condition. (Program control cannot be transferred to a register.) Register deferred mode is legal and will cause program control to be transferred to the address held in the specified register. Note that instructions are word data and must therefore be fetched from an even numbered address. A boundary error trap condition will result when the processor attempts to fetch an instruction from an odd address.

Table 2-4 Program Control Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
JRS Jump to subroutine	004RDD	(tmp) ← (dst) (tmp is an internal processor register) ↓ (SP) ← reg (push reg contents onto processor stack) reg ← PC PC holds location following JSR; this address PC ← (tmp), now put in (reg)	Unaffected	In execution of the JSR, the old contents of the specified register (the linkage pointer) are automatically pushed onto the processor stack and new linkage information placed in the register. Thus, subroutines nested within subroutines to any depth may all be called with the same linkage register. There is no need either to plan the maximum depth at which any particular subroutine will be called or to include instructions in each routine to save and restore the linkage pointer. Further, since all linkages are saved in a re-entrant manner on the processor stack, execution of a subroutine may be interrupted, and the same subroutine re-entered and executed by an interrupt service routine. Execution of the initial subroutine can then be resumed when other requests are satisfied. This process (called nesting) can proceed to any level. JSR PC, dst is a special case of the PDP-11 subroutine call suitable for subroutine calls that transmit parameters.
RTS Return from subroutine	00020R	PC ← (reg) (reg) ← SP ↑	Unaffected	Loads contents of register into PC and pops the top element of the processor stack into the specified register. Return from a non-re-entrant subroutine is typically made through the same register that was used in its call. Thus, a subroutine called with a JSR PC, dst exits with an RTS PC, and a subroutine called with a JSR R5, dst may pick up parameters with addressing modes (R5) +, X (R5), or @X (R5) and finally exit, with an RTS R5.

Table 2-4 Program Control Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description

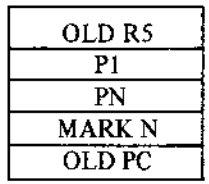
MARK	0064NN	SP ← SP + 2xnn
		PC ← R5
		R5 ← (SP) ↑
		nn = number of parameters
			Unaffected	Used as part of the standard PDP-11 subroutine return convention. MARK facilitates the stack cleanup procedures involved in subroutine exit. Assembler format is: MARK N
				
				Example: MOV R5,-(SP) ;place old R5 on stack
				MOV P1,-(SP) ;place N parameters on
				MOV P2,-(SP) ;the stack to be used
				;there by the subroutine
				
				MOV PN,-(SP) ;places the instruction
				MOV #MARKN,-(SP) ;MARK N on the stack
				;set up address at Mark
				MOV SP,R5 ;N instruction
				
				JSR PC,SUB ;jump to subroutine
				
				At this point the stack is as follows:
				
				 
				
				And the program is at the address SUB which is the beginning of the subroutine.
				SUB: ;execution of the subroutine itself
				RTS R5: ;the return begins
				
				This causes the contents of R5 to be placed in the PC which then results in the execution of the instruction MARK N. The contents of old PC are placed in R5
				
				MARK N causes: (1) the stack pointer to be adjusted to point to the old R5 value; (2) the value now in R5 (the old PC) to be placed in the PC; and (3) contents of the old R5 to be popped into R5, thus completing the return from subroutine.

Table 2-4 Program Control Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
SOB Subtract one and branch if not equal to 0	077R00 plus offset	R ← R - 1 if this result ≠ 0 then PC ← PC - (2 x offset)	Unaffected	The register is decremented. If it is not equal to 0, twice the offset is subtracted from the PC (now pointing to the following word). The offset is interpreted as a six-bit positive number. This instruction provides a fast, efficient method of loop control. Assembler syntax is: SOB R,A where A is the address to which transfer is to be made if the decremented R is not equal to 0. Note that the SOB instruction cannot be used to transfer control in the forward direction.
BPT Break-point Trap	000003	↓ (SP) ← PS ↓ (SP) ← PC PC ← (14) PS ← (16)	N: loaded from trap vector Z: loaded from trap vector V: loaded from trap vector C: loaded from trap vector	Performs a trap sequence with a trap vector address of 14. Used to call debugging aids. The user is cautioned against employing code 000003 in programs run under these debugging aids.
IOT IOT Trap	000004	↓ (SP) ← PS ↓ (SP) ← PC PC ← (20) PS ← (22)	N: loaded from trap vector Z: loaded from trap vector C: loaded from trap vector	Performs a trap sequence with a trap vector address of 20. Used to call the I/O executive routine IOX in the paper-tape software system and for error reporting in the disk operating system.

Table 2-4 Program Control Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
EMT Emulator Trap	104000	↓ (SP) ← PS ↓ (SP) ← PC PC ← (30) PS ← (32)	N: loaded from trap vector Z: loaded from trap vector V: loaded from trap vector C: loaded from trap vector	All operation codes from 104000 to 104377 are EMT instructions and may be used to transmit information to the emulating routine (e.g., function to be performed). The trap vector for EMT is at address 30; the new central processor status (PS) is taken from the word at address 32. **CAUTION** **EMT is used frequently by DEC system software and is therefore not recommended for general use.**
TRAP	104400 to 104777	↓ (SP) ← PS ↓ (SP) ← PC PC ← (34) PS ← (36)	N: loaded from trap vector Z: loaded from trap vector V: loaded from trap vector C: loaded from trap vector	Operation codes from 104400 to 104777 are TRAP instructions. TRAPs and EMTs are identical in operation, except that the trap vector for TRAP is at address 34. **NOTE** **Since DEC software makes frequent use of EMT, the TRAP instruction is recommended for general use.**

NOTE: Condition Codes are unaffected by these instructions

*DD = destination (address mode and register)
† (dst) = destination contents

Table 2-5 Miscellaneous Instructions

Mnemonic	OP Code	Operation	Condition Codes	Description
RTI	000002	PC ← (SP) ↑ PSW ← (SP) ↑	N: loaded from processor stack Z: loaded from processor stack V: loaded from processor stack C: loaded from processor stack	Used to exit from an interrupt or trap service routine. The PC and PSW are restored (popped) from the processor stack. If the RTI sets the T-bit in the PSW, a trace trap will occur prior to executing the next instruction.
RTT	000006	PC ← (SP) ↑ PS ← (SP) ↑	N: loaded from processor stack Z: loaded from processor stack V: loaded from processor stack C: loaded from processor stack	This is the same as the RTI instruction, except that it inhibits a trace trap, while RTI permits a trace trap. If a trace trap is pending, the first instruction after the RTT will be executed prior to the next "T" trap. In the case of the RTI instruction, the "T" trap will occur immediately after the RTI.
MFPI MFPD	0065SS 1065SS	(temp) ← (src) ↓ (SP) ← (temp)	N: set if the source <0; otherwise cleared Z: set if the source = 0; otherwise cleared V: cleared C: unaffected	This instruction pushes a word onto the current stack from an address in previous space. Processor Status (bits 13, 12). The source address is computed using the current registers and memory map.
MTPI MTPD	0066SS 1066SS	(temp) ← (SP) ↑ (dst) ← (temp)	N: set if the source <0; otherwise cleared Z: set if the source = 0; otherwise cleared V: cleared C: unaffected	This instruction pops a word off the current stack determined by PS (bits 15, 14) and stores that word into an address in previous space PS (bits 13, 12). The destination address is computed using the current registers and memory map.

Table 2-5 Miscellaneous Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
MFPS	1067DD	(DST) ← PSW DST Lower 8 bits	N: set if PSW bit 7=1; otherwise cleared. Z: set if PS[0:7]=0; otherwise cleared. V: cleared C: not affected	*The 8-bit contents of the PS are moved to the effective destination. If destination is mode 0, PS bit 7 is sign-extended through upper byte of the register, and destination operand is treated as a byte address.
MTPS	1064SS	PSW ← (SRC)	Set according to effective SRC operand 0–3.	*The 8 bits of the effective operand replace the current contents of the PSW. The source operand address is treated as a byte address. Note that PSW bit 4 cannot be set with this instruction. The SRC operand remains unchanged. *Because there is no hardware to prevent execution of these instructions in User mode, it is necessary for the system software to prevent any reference to the PSW address by a user.
HALT	000000		Unaffected	Causes the processor operation to cease. The console is given control of the processor. The console data lights display the address of the HALT instruction plus two. Transfers on the Unibus are terminated immediately. The PC points to the next instruction to be executed. Pressing the CON key on the console causes processor operation to resume. No INIT signal is given.

Table 2-5 Miscellaneous Instructions (Cont)

Mnemonic	OP Code	Operation	Condition Codes	Description
WAIT	000001		Unaffected	Provides a way for the processor to relinquish use of the bus while it waits for an external interrupt. Having been given a WAIT command, the processor will not compete for bus by fetching instructions or operands from memory. This permits higher transfer rates between device and memory, as no processor-induced latencies will be encountered by bus requests from the device. In WAIT, as in all instructions, the PC points to the next instruction following the WAIT operation. Thus, when an interrupt causes the PC and PS to be pushed onto the stack, the address of the next instruction following the WAIT is saved. The exit from the interrupt routine (i.e., execution of an RTI instruction) will cause resumption of the interrupted process at the instruction following the WAIT.
RESET	000005	PC (SP) PSW (SP)	Unaffected	Sends INIT on the Unibus for 100 ms. All devices on the Unibus are reset to their state at power-up.

Table 2-6 Condition Code Operators

Mnemonic	Op Code	Instruction
CLC	000241	Clear condition code C.
CLV	000242	Clear condition code V.
CLZ	000244	Clear condition code Z.
CLN	000250	Clear condition code N.
CCC	000257	Clear all condition code bits.
SEC	000261	Set condition code C.
SEV	000262	Set condition code V.
SEZ	000264	Set condition code Z.
SEN	000270	Set condition code N.
SCC	000277	Set all condition code bits.

NOTE

Selectable combinations of condition code bits may be cleared or set together. The status of bit 4 controls the way in which bits 0, 1, 2, and 3 are to be modified. If bit 4 = 1, the specified bits are set; if bit 4 = 0, the specified bits are cleared.

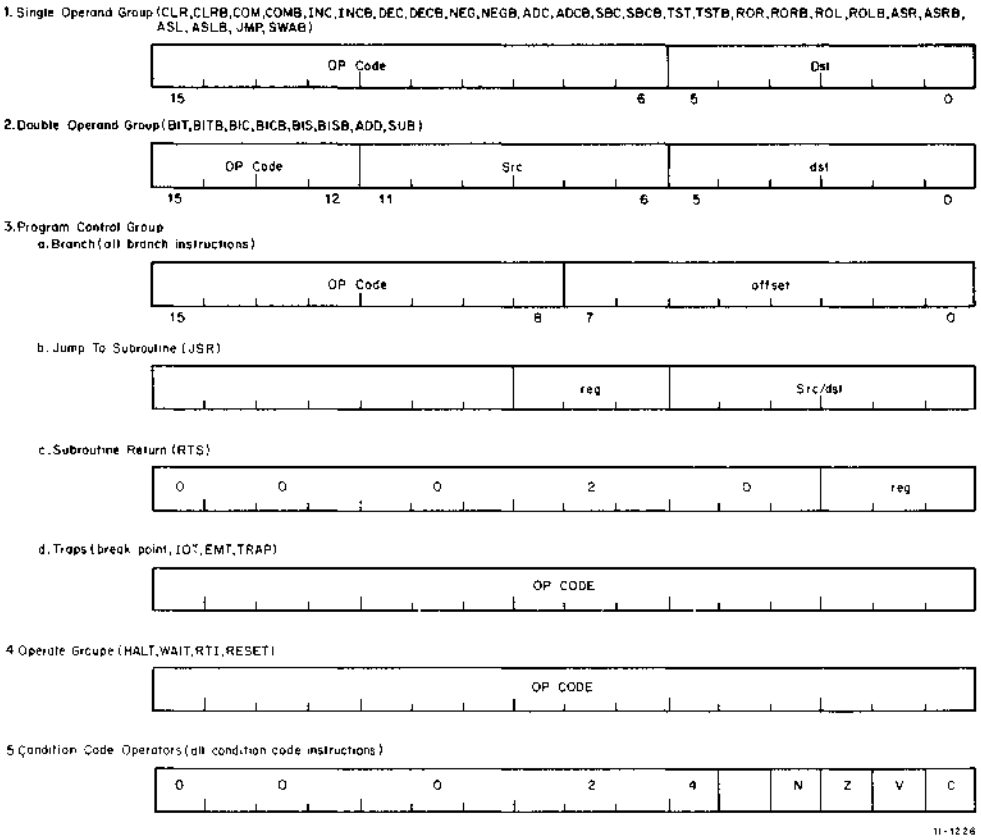


Figure 2-2 PDP-11 Instruction Formats

2.4 INSTRUCTION EXECUTION TIME

The execution time for an instruction depends on the instruction itself, the modes of addressing used, and the type of memory being referenced. In the most general case, the instruction execution time is the sum of a source address (SRC) time, a destination address (DST) time, and an execute, fetch (EF) time.

Instr Time = SRC Time + DST Time + EF Time

Some of the instructions require only some of these times, and are so noted in Paragraph 2.4.1. All timing information is in microseconds, unless otherwise noted. Times are typical; processor timing can vary ±10%.

2.4.1 Basic Instruction Set Timing

Table 2-7 lists the PDP-11/34 instruction set, together with the timing characteristics and memory cycles required. The timing requirements for determining instruction execution time are listed below.

Double-Operand (all instructions)

Instr Time = SRC Time + DST Time + EF Time

Single-Operand (all instructions)

Instr Time = DST Time + EF Time

Branch, Jump, Control, Trap, and Miscellaneous (all instructions)

Instr Time = EF Time

NOTES

1. **The times specified apply to both word and byte instructions, whether odd or even byte.**
2. **Timing is given without regard for NPR or BR servicing.**
3. **If the memory management is enabled, instruction execution times increase by 0.12 μs for each memory cycle used.**
4. **All timing is based on memory with the following performance characteristics:**

Memory	Access Time (μs)	Cycle Time (μs)
Core (MM11-DP)	0.510	1.1
MOS (MS11-JP)	0.635	0.920

Table 2-7 PDP-11/34 Instruction Set

SOURCE ADDRESS TIME

Instruction	Source Mode	Memory Cycles	Core (MM11-DP) μs	MOS (MS11-JP) μs
Double Operand	0	0	0.00	0.00
	1	1	1.13	1.26
	2	1	1.33	1.46
	3	2	2.37	2.62
	4	1	1.28	1.41
	5	2	2.57	2.82
	6	2	2.57	2.82
	7	3	3.80	4.18

DESTINATION TIME

Instruction	Destination Mode	Memory Cycles	Core	MOS
Modifying Single-Operand and Modifying Double-Operand (Except MOV, SWAB, ROR, ROL, ASR, ASL)	0	0	0.00	0.00
	1	2	1.62	1.74
	2	2	1.77	1.89
	3	3	2.90	3.15
	4	2	1.77	1.89
	5	3	3.00	3.25
	6	3	3.10	3.35
	7	4	4.29	4.66
MOV	0	0	0.00	0.00
	1	1	0.93	0.93
	2	1	0.93	0.93
	3	2	2.17	2.29
	4	1	1.13	1.13
	5	2	2.22	2.34
	6	2	2.37	2.49
	7	3	3.50	3.75
MTPS	0	0	0.00	0.00
	1	1	0.95	0.95
	2	1	1.13	1.26
	3	2	2.26	2.51
	4	1	1.13	1.26
	5	2	2.26	2.51
	6	2	2.44	2.69
	7	3	3.57	4.20
MFPS	0	0	0.00	0.00
	1	1	0.64	0.64
	2	1	0.64	0.64
	3	2	1.95	2.08
	4	1	0.82	0.82
	5	2	1.95	2.08
	6	2	2.13	2.26
	7	3	3.26	3.51

Table 2-7 PDP-11/34 Instruction Set (Cont)

EXECUTE, FETCH TIME

Instruction	Destination Mode	Memory Cycles	Core	MOS
Double Operand				
ADD, SUB, CMP, BIT, BIC, BIS, XOR		1	2.03	2.16
MOV		1	1.83	1.96
Single Operand				
CLR, COM, INC, DEC, ADC, SBC, TST		1	1.83	1.96
SWAB, NEG		1	2.03	2.16
ROR, ROL, ASR, ASL		1	2.18	2.31
MTPS		2	2.99	3.12
MFPS		2	1.99	2.12
EIS Instructions (use with DST times)				
MUL		1	8.82*	8.95*
DIV (overflow)		1	2.78	2.91
			12.48	12.61
ASH		1	4.18**	4.31**
ASHC		1	4.18**	4.31**
Memory Management Instructions				
MFPI(D)		2	3.07	3.14
MTPI(D)		2	3.37	3.34
SWAB, ROR, ROL, ASR, ASL	0	0	0.00	0.00
	1	2	1.42	1.54
	2	2	1.57	1.69
	3	3	2.70	2.95
	4	2	1.62	1.74
	5	3	2.80	3.05
	6	3	2.90	3.15
	7	4	4.09	4.46
Non-modifying Single Operand and Double Operand	0	0	0.00	0.00
	1	1	1.13	1.26
	2	1	1.28	1.41
	3	2	2.42	2.67
	4	1	1.33	1.46
	5	2	2.52	2.77
	6	2	2.62	2.87
	7	3	3.80	4.18

Table 2-7 PDP-11/34 Instruction Set (Cont)

EXECUTE, FETCH TIME

Instruction	Destination Mode	Memory Cycles	Core	MOS
MFPI(D)	0	0	0.00	0.00
MTPI(D)	1	1	0.98	1.24
	2	1	1.32	1.44
	3	2	2.20	2.45
	4	1	1.18	1.44
	5	2	2.20	2.45
	6	2	2.40	2.65
	7	3	3.59	3.96
Branch Instructions				
BR, BNE, BEQ, (Branch) BPL, BMI, BVC, BVS, BCC, BCS, BGE, BLT, BGT, BLE, BHI, BLOS, BHIS, BLO		1	2.18	2.31
(No Branch)		1	1.63	1.76
SOB (Branch)		1	2.38	2.51
(No Branch)		1	1.98	2.11
Jump Instructions				
JMP	1	1	1.83	1.96
	2	1	2.18	2.31
	3	2	3.12	3.37
	4	1	2.03	2.16
	5	2	3.07	3.32
	6	2	3.07	3.32
	7	3	4.25	4.78
JSR	1	2	3.32	3.44
	2	2	3.47	3.59
	3	3	4.40	4.65
	4	2	3.32	3.44
	5	3	4.40	4.65
	6	3	4.60	4.85
	7	4	5.69	6.06
RTS		2	3.32	3.57
MARK		2	4.27	4.52
RTI, RTT		3	4.60	4.98
Set or Clear C, V, N, Z		1	2.03	2.16
HALT		1	1.68	1.81
WAIT		1	1.68	1.81
RESET		1	100 ms	100 ms
IOT, EMT, TRAP, BPT		5	7.32	7.7

*Add 200 ns for each bit transition in serial data from LSB to MSB.
**Add 200 ns per shift.

2.4.2 Bus Latency Times

Interrupts (BR requests) are acknowledged at the end of the current instruction. For a typical instruction, with an instruction execution time of 4 μs, the average time to request acknowledgement would be 2 μs.

Interrupt service time, which is the time from BR acknowledgement to the first subroutine instruction, is 7.32 μs max for core, and 7.7 μs for MOS.

NPR (DMA) latency, which is the time from request to bus mastership for the first NPR device, is 2.5 μs max.

2.5 EXTENDED INSTRUCTION SET

The Extended Instruction Set (EIS) provides the user with the capability of extended manipulation of fixed-point numbers. Use of the EIS instructions does not degrade processor timing or affect NPR latency. Interrupts are serviced at the end of an EIS instruction.

The EIS instructions are:

Mnemonic	Instruction	Op Code
MUL	Multiply	070RSS
DIV	Divide	071RSS
ASH	Shift arithmetically	072RSS
ASHC	Arithmetic shift combined	073RSS

The number formats are shown in Figure 2-3. Examples of the operation of each instruction are presented in the paragraphs that follow.

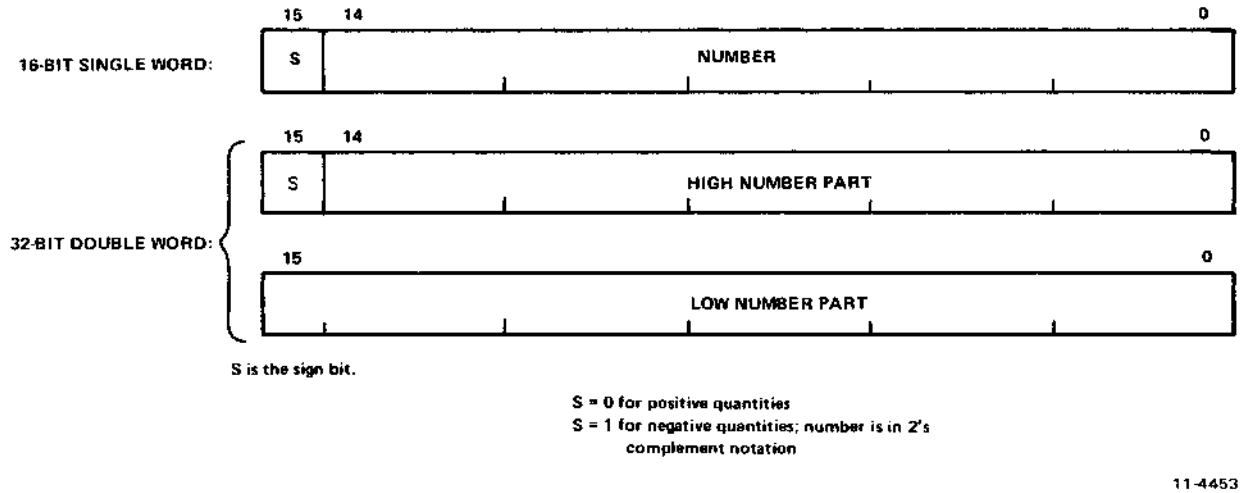


Figure 2-3 Extended Instruction Set Number Formats

Multiply Instruction - MUL 070RSS

Example: 16-bit product (R is odd)

```
000241          , CLC              ;Clear carry condition code
012701,400      , MOV #400,R1
070127,10       , MUL #10, R1
1034xx          , BCS ERROR        ;Carry will be set if
                                   ;product is less than
                                   ;-2^15 or greater than or ;equal to 2^15
                                   ;no significance lost
```

Before	After
(R1) = 000400	(R1) = 004000

Divide Instruction - DIV 071RSS

Example:

```
005000          , CLR R0
012701,20001    , MOV #20001,R1
071027,2        , DIV #2,R0
```

Before	After
(R0) = 000000	(R0) = 010000 Quotient
(R1) = 020001	(R1) = 000001 Remainder

Arithmetic Shift Instruction - ASH 072RSS

Example: ASH R0, R3

Before	After
(R3) = 000003	(R3) = 000003
(R0) = 001234	(R0) = 012340

Arithmetic Shift Combined Instruction - ASHC 073RSS

Example: Similar to the example for the ASH instruction except that two registers are used.

2.6 INSTRUCTION SET DIFFERENCES

Table 2-8 lists the instruction set differences between the PDP-11/34 and other PDP-11 machines.

Table 2-8 Programming Differences

	11/05 and 11/10	11/35 and 11/40	11/04	11/34
GENERAL REGISTERS (including PC and SP)				
OPR%R,(R)+ or OPR%R,-(R) OPR%R,@(R)+ OPR%R,@-(R) (Using the same register as both source and destination)	Initial contents of R are used as the source operand.	Contents of R are incremented by 2 (or decremented by 2), before being used as the source operand.	Same as 11/05	Same as 11/05
JMP(R)+ or JSR register, (R)+ (jump using autoincrement)	Contents of R are incremented by 2, then used as the new PC address.	Initial contents of P are used as new PC.	Same as 11/40	Same as 11/40
MOV PC, @#A or MOV PC, A (Moving the incremented PC to a memory address referenced by the PC)	Location A will contain PC + 2.	Location A will contain the PC of the move instruction +4.	Same as 11/05	Same as 11/05
Stack Pointer (SP), R6 used for referencing.	Using the SP for pointing to odd addresses or nonexistent memory causes a halt (double bus error).	Odd address or nonexistent memory references with SP cause a fatal trap with a new stack created at locations 0 and 2.	Same as 11/05	Same as 11/05

Table 2-8 Programming Differences (Cont)

	11/05 and 11/10	11/35 and 11/40	11/04	11/34
GENERAL REGISTERS (including PC and SP) (Cont)				
Stack Overflow	Stack limit fixed at 400_8 overflow (going lower) checked after modes 4 and 5 using R6, and JSR and traps. Overflow serviced by an overflow trap. No red zone.	Variable limit with stack limit option. Overflow checked after JSR, traps, and address modes 1, 2, 4, and 6. Non-altering references to stack data are always allowed. There is a 16-word yellow (warning) zone. Red zone trap occurs if stack is 16 words below boundary; PS and PC are saved at locations 0 and 2.	Same as 11/05	Same as 11/05
TRAPS AND INTERRUPTS				
RTI Instruction	First instruction after RTI instruction is always executed.	If RTI sets the T-bit, the T-bit trap is acknowledged immediately after the RTI instruction.	Same as 11/40	Same as 11/40
RTT Instruction	Not implemented	First instruction after RTT is guaranteed to be executed.	Same as 11/40	Same as 11/40
Processor status odd byte at location 777777	Odd byte of PS can be addressed without a trap.	Same as 11/05	Same as 11/05	Same as 11/05

Table 2-8 Programming Differences (Cont)

	11/05 and 11/10	11/35 and 11/40	11/04	11/34
GENERAL REGISTERS (including PC and SP) (Cont)				
T-bit of PS	T-bit can be loaded by direct address of PS or from console.	Only RTI, RTT traps and interrupts can load the T-bit	Same as 11/05	Same as 11/40
Bus Errors				
PC contains odd address	PC unincremented	Same as 11/05	Same as 11/05	Same as 11/05
PC contains an address in nonexistent memory	PC incremented	PC unincremented	Same as 11/05	Same as 11/40
Register contains odd address and instruction mode 2	Register unincremented	Register incremented	Same as 11/05	Same as 11/05 except for MOV mode 2 and MTPI where the register will be incremented.
Register contains address in nonexistent memory and instruction mode 2.	Register incremented.	Register incremented.	Register unincremented	Same as 11/04 except for MOV mode 2 destination and MTPI where the register will be incremented.
Interrupt service routine.	The first instruction will not be executed if another interrupt occurs at a higher priority.	Same as 11/05	Same as 11/05	Same as 11/05

Table 2-8 Programming Differences (Cont)

	11/05 and 11/10	11/35 and 11/40	11/04	11/34
GENERAL REGISTERS (including PC and SP) (Cont)				
Priority order of of traps and interrupts	Odd address Time-out Halt instruction Trap instructions Trace trap Stack overflow Power fail Halt from console	Halt instruction Odd address Stack overflow (red) Mem mgt error Time-out Parity Trap instruction Trace trap Stack overflow (yellow) Power fail Halt from console	Halt instruction Bus error Trap instruction Trace trap Stack overflow Power fail Halt from console Interrupts Next instruction fetch	Same as 11/40 except no red zone stack overflow
MISCELLANEOUS				
Swab and V-bit	V-bit is cleared.	Same as 11/05	Same as 11/05	Same as 11/05
Instruction set	Basic set	Basic set and Mark, RTT, SOB, SxT, XOR. EIS adds: MUL, DIV ASH, ASHC. FIS adds: FADD, FSUB, FMUL, FDIV. KT11-D adds: MTPI, MFPI.	Basic set and RTT	Basic set and Mark, RTT, SOB, SxT, XOR, MUL, DIV, ASH, ASHC, MTPI, MFPI, MTPS, MFPS. (MTPS and MFPS are new instructions used for LSI-11.)
Memory management violation during a trap sequence	Does not apply	If a mem mgt violation occurs between the first and second push down of the stack during a trap sequence, the status of the CPU before the violation is placed as the PS on the the Kernel stack.	Does not apply	If a mem mgt violation occurs between the first and second push down of the stack during a trap sequence, the status of the vector +2 of the original trap is placed as the PS on the Kernel stack.

CHAPTER 3
CPU OPERATING SPECIFICATIONS

Operating Temperature	5° to 50° C (41° to 122° F)
Relative Humidity	20 to 95% (without condensation)
Input Power	+5 Vdc ±5% at 4.5 A (typical) per module (M7265 and M7266)
Physical Size	Two hex modules (8-1/2 × 15 in.)
Interface Requirements	All I/O signals are available on connectors A and B. These signals are pin-compatible with modified Unibus pinout as shown in Table 3-1. The bus loading on each of these Unibus lines is equivalent to one bus load.
Power and Ground Pinouts	+5 V: pins AA2, BA2, CA2, DA2, EA2, EA2 GND: pins AC2, AT1, BC2, BT1, CC2, CT1, DC2, DT1, EC2, ET1, FC2, FT1
Number of Integrated Circuits	231 (M7265 = 120; M7266 = 111)

Table 3-1 Standard and Modified Unibus Pin Assignments

Pin	Standard Signal	Modified Signal	Pin	Standard Signal	Modified Signal	Pin	Standard Signal	Modified Signal
AA1	INIT L	INIT L	AP1	GROUND	P0*	BH1	A01 L	A01 L
AA2	+5 V	+5 V	AP2	BBSY L	BBSY L	BH2	A00 L	A00 L
AB1	INTR L	INTR L	AR1	GROUND	BAT BACKUP +15 V	BJ1	A03 L	A03 L
AB2	GROUND	TEST POINT	AR2	SACK L	SACK L	BJ2	A02 L	A02 L
AC1	D00 L	D00 L	AS1	GROUND	BAT BACKUP +15 V	BK1	A05 L	A05 L
AC2	GROUND	GROUND	AS2	NPR L	NPR L	BK2	A04 L	A04 L
AD1	D02 L	D02 L	AT1	GROUND	GROUND	BL1	A07 L	A07 L
AD2	D01 L	D01 L	AT2	BR7 L	BR7 L	BL2	A06 L	A06 L
AE1	D04 L	D04 L	AU1	NPG H	+20 V	BM1	A09 L	A09 L
AE2	D03 L	D03 L	AU2	BR6 L	BR6 L	BM2	A08 L	A08 L
AF1	D06 L	D06 L	AV1	BG7 H	+20 V	BN1	A11 L	A11 L
AF2	D05 L	D05 L	AV2	GROUND	+20 V	BN2	A10 L	A10 L
AH1	D08 L	D08 L	BA1	BG6 H	SPARE	BP1	A13 L	A13 L
AH2	D07 L	D07 L	BA2	+5 V	+5 V	BP2	A12 L	A12 L
AJ1	D10 L	D10 L	BB1	BG5 H	SPARE	BR1	A15 L	A15 L
AJ2	D09 L	D09 L	BB2	GROUND	TEST POINT	BR2	A14 L	A14 L
AK1	D12 L	D12 L	BC1	BR5 L	BR5 L	BS1	A17 L	A17 L
AK2	D11 L	D11 L	BC2	GROUND	GROUND	BS2	A16 L	A16 L
AL1	D14 L	D14 L	BD1	GROUND	BAT BACKUP +5 V	BT1	GROUND	GROUND
AL2	D13 L	D13 L	BD2	BR4 L	BR4 L	BT2	C1 L	C1 L
AM1	PA L	PA L	BE1	GROUND	INT SSYN*	BU1	SSYN L	SSYN L
AM2	D15 L	D15 L	BE2	BG4 H	PAR: DET*	BU2	C0 L	C0 L
AN1	GROUND	P1*	BF1	ACLO L	ACLO L	BV1	MSYN L	MSYN L
AN2	PB L	PB L	BF2	DCLO L	DCLO L	BV2	GROUND	-5 V

*Pins used by parity control module.

CHAPTER 4
DETAILED HARDWARE DESCRIPTION

4.1 INTRODUCTION

The following paragraphs contain a detailed circuit description of the KD11-E Central Processing Unit (CPU), which is used in the PDP-11/34 Computer. Segments of the CPU, shown in Figure 4-1, are analyzed separately, using the block diagrams contained in this manual and the KD11-E circuit schematics.

4.2 DATA PATH

4.2.1 General Description

The simplified KD11-E data path consists of six function units, as shown in Figure 4-2. Circuit schematics K1-1 through K1-4 (D-CS-M7265-0-1) each contain one 4-bit slice of the data path. Table 4-1 briefly describes the function of each of the six function units.

Data flow through the data path is controlled, directly or indirectly, by the Control Store circuitry on the control module (M7266). Each Control Store ROM location (microinstruction) generates a unique set of outputs capable of controlling the data path elements and determining the ALU function to be performed. Sequences of these ROM microinstructions are combined into microroutines, which perform the various PDP-11 instruction operations.

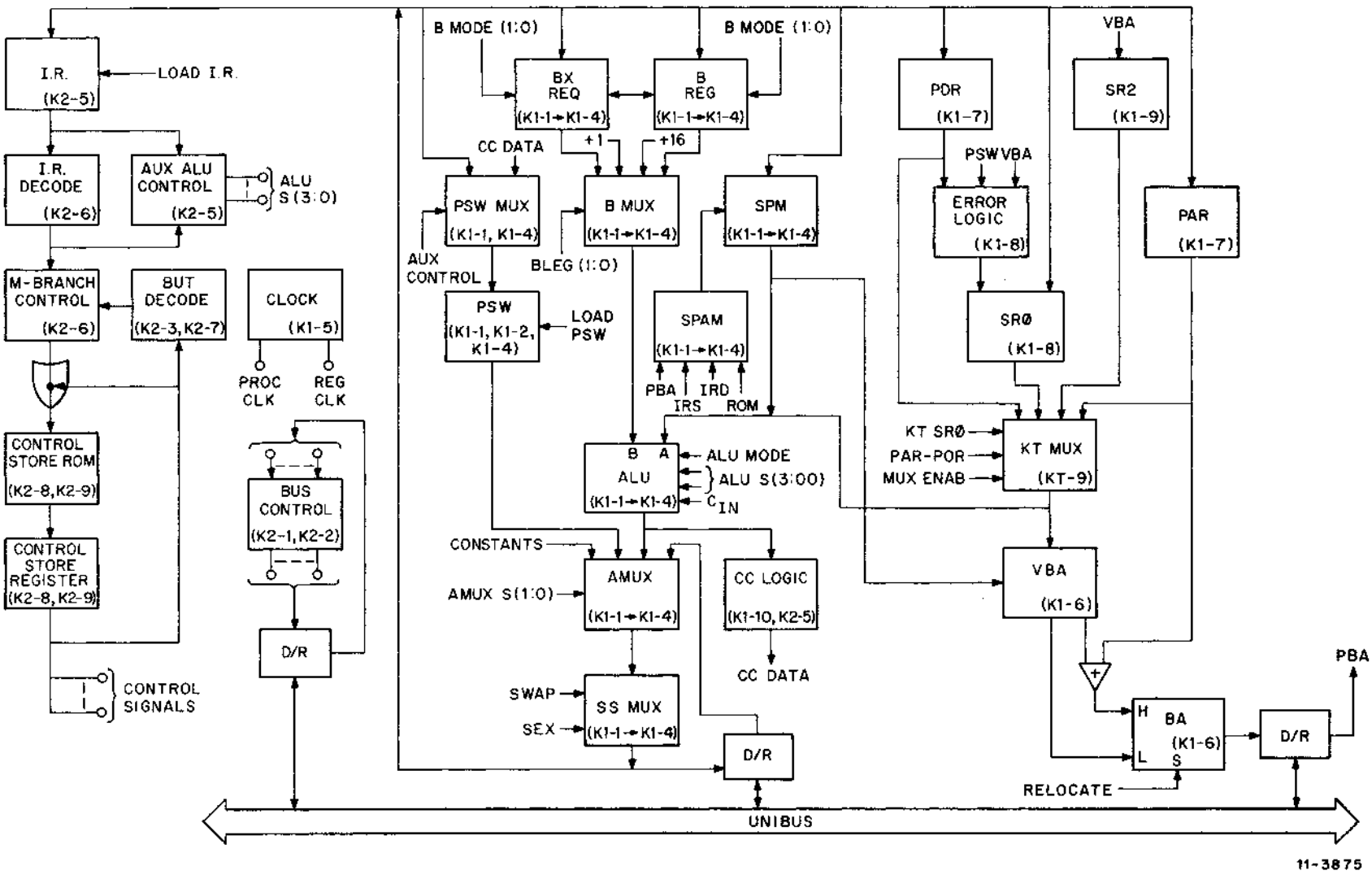


Figure 4-1 KD11-E Block Diagram

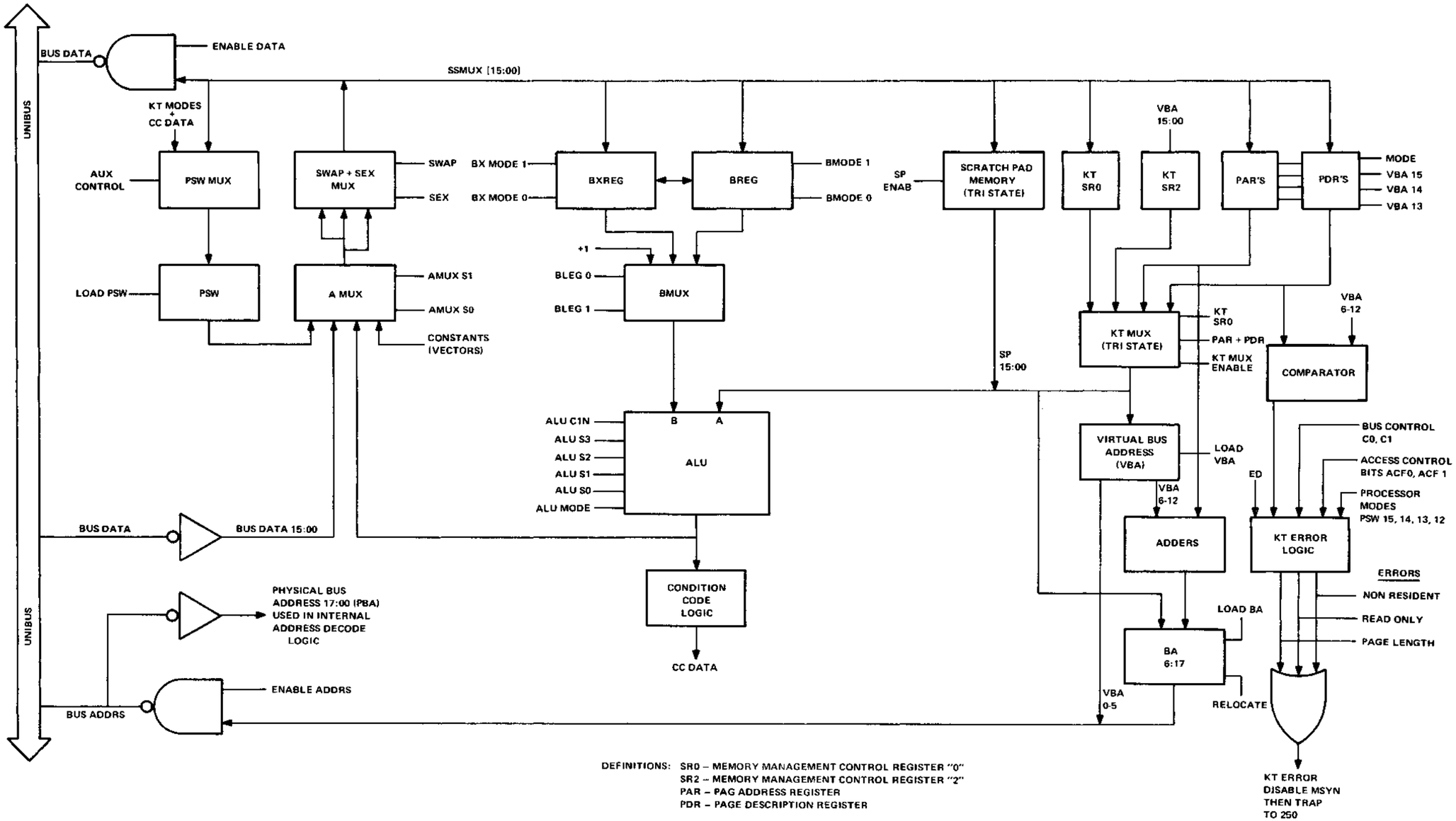


Figure 4-2 Simplified KD11-E Data Path

Table 4-1 Function Units of the KD11-E Data Path

Unit	Function
Arithmetic Logic Unit (ALU)	The heart of the data path is the ALU, which is the logic element that manipulates the data. It is capable of performing 16 arithmetic or 16 logic (Boolean) operations on two 16-bit operands to produce a 16-bit result. The A input comes from either the scratchpad memory or the memory management system; the B input comes from the B leg. The ALU output is sent to the AMUX.
ALU Multiplexer (AMUX)	The AMUX is a 4-to-1 multiplexer that controls the introduction of new data and the circulation of available data through the data path. Input to the AMUX is both external (from the Unibus data lines) and internal (from the ALU, PSW, or constants). The AMUX output is sent to the SSMUX.
Processor Status Word Register (PSW)	The PSW register is a 12-bit register that contains information on the current processor priority, condition codes (C, V, Z, and N) which indicate the results of the last instruction, a "trap" bit (TBIT) which causes automatic traps after each fetch instruction used during program debugging, and both the current and previous memory management modes (Kernel or User). PSW input comes from the SSMUX or from condition code logic; PSW output is sent to the AMUX.
Swap Sign Extend Multiplexer (SSMUX)	This multiplexer controls the form in which data is output from, or recirculated into, the data path. The SSMUX can pass the data unchanged, swap the high and low bytes, sign-extend the low byte into the entire word, or simultaneously swap high and low bytes while sign-extending the high byte (which becomes the new low byte) into the entire word. SSMUX input comes from the AMUX. SSMUX output goes to either the rest of the computer system (via the Unibus), the other sections of the processor (the control section, via the Instruction register, and the memory management system), or to other portions of the data path (the PSW, the B leg, and the scratchpad memory).
B Leg	The B leg of the ALU consists of two 16-bit registers (B and BX) and a 4-to-1 multiplexer (BMUX). Both registers can shift left or right independently, or together they can perform full 32-bit shifts. The BMUX selects one of the four functions (BREG, BXREG, +1, +16) and connects to the B input of the ALU. The B leg is used to store operands for the ALU, to implement rotate and shift instructions, and to implement Extended Instruction Set (EIS) instructions. B leg input comes from the SSMUX. B leg output goes to the B input of the ALU.

Table 4-1 Function Units of the KD11-E Data Path (Cont)

Unit	Function
Scratchpad Memory (SPM)	This random access memory can store sixteen 16-bit words in eight processor-dedicated registers and eight general-purpose (user available) registers. One of the general-purpose registers is used as a stack pointer, another as the program counter. Input to the scratchpad memory is from the SSMUX. Output, which can be buffered and latched to enable reading from one address and modifying another during the same cycle, goes to the A input of the ALU and to the Virtual and Physical Bus Address registers.

4.2.2 Arithmetic Logic Unit (ALU)

The ALU (Figure 4-3) is divided into four 4-bit slices (K1-1, K1-2, K1-3, and K1-4 each contain a slice), with each slice consisting of one 4-bit ALU chip (74S181) and part of a Look-Ahead Carry Generator chip (74S182).

ALU Inputs

The A input to each ALU chip comes from one of the scratchpad memory (SPM) registers or from the KTMUX, as specified by the Control Store microinstruction being performed. (Refer to Paragraph 4.2.3 for details.) The B input comes from the B leg multiplexer (BMUX) logic, and can take the form of the B register contents, the BX register contents, a constant 0, a constant 1, or a constant 16. (Refer to Paragraph 4.2.4 for details.)

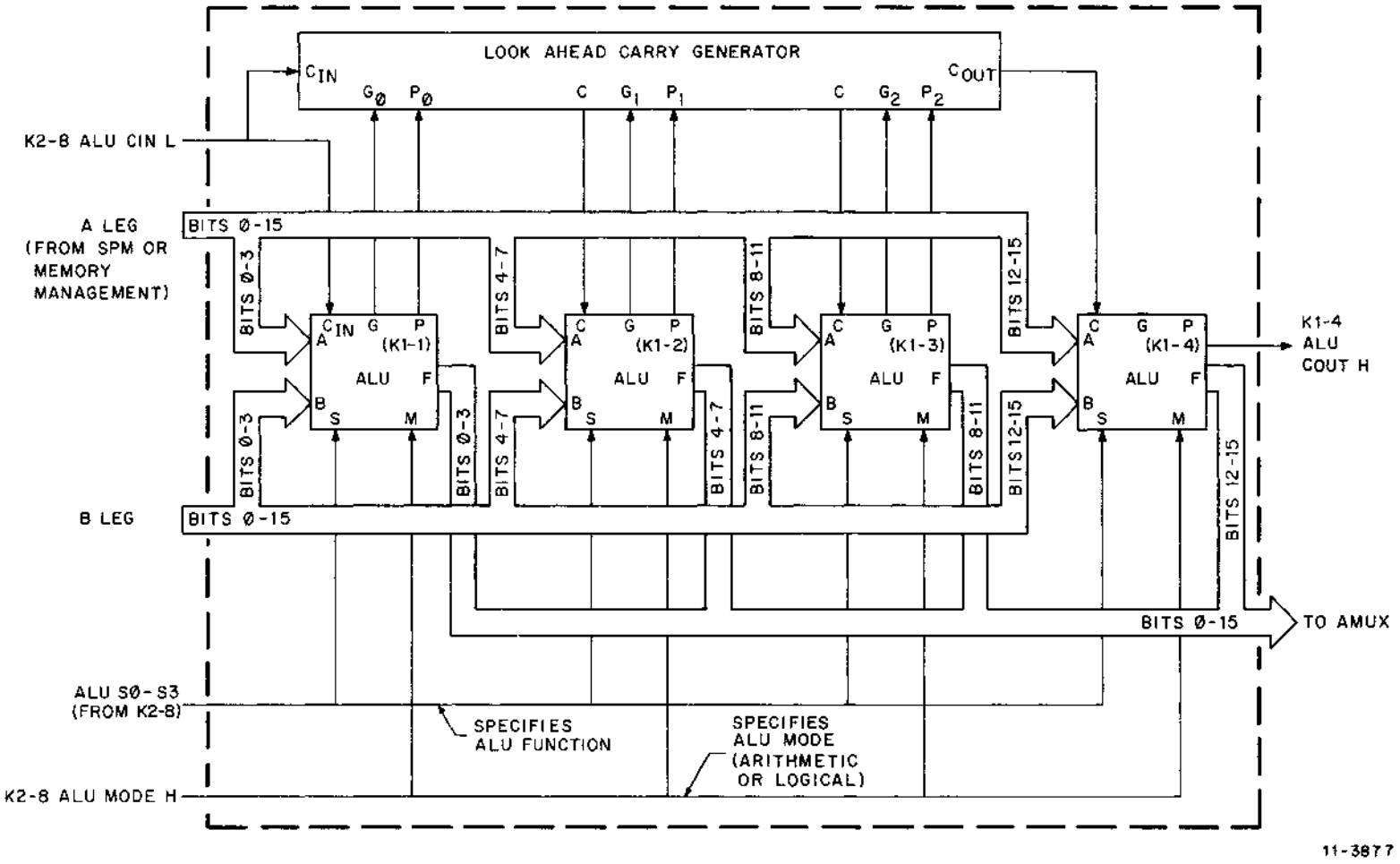


Figure 4-3 ALU Block Diagram

ALU Functions
The function performed by the ALU is controlled by the four Selection bits (S3, S2, S1, S0), the Mode bit (M), and the Carry-In bit (CIN). Table 4-2 lists the ALU functions of the KD11-E and the corresponding bit patterns for the six control signals.

Table 4-2 ALU Functions and Control Signals

ALU Function	ALU Control Signals					
	S3	S2	S1	S0	CIN	M
$\overline{\text{ZERO}}$	0	0	1	1	0	1
$\overline{\text{A}}$	0	0	0	0	0	1
A plus 1	0	0	0	0	0	0
A minus 1	1	1	1	1	1	0
A minus B	0	1	1	0	0	0
A	1	1	1	1	0	1
B	1	0	1	0	0	1
A plus B	1	0	0	1	1	0
$\overline{\text{A}}$·B	1	0	1	1	0	1
$\overline{\text{A}}$·B	0	0	1	0	0	1
A plus B plus 1	1	0	0	0	0	0
$\overline{\text{A}}$ plus A	1	1	0	0	1	0
$\overline{\text{B}}$	0	1	0	1	0	1
A plus A plus 1	1	1	0	0	0	0
A$\oplus$B	0	1	1	0	0	1

4.2.3 Scratchpad
The scratchpad consists of a random access memory that can store sixteen 16-bit words, and can be used for various functions. Scratchpad operation is divided into four 4-bit slices, with K1-1, K1-2, K1-3, and K1-4 (D-CS-M7265-0-1) each containing one slice. The scratchpad address multiplexer circuitry is shown on K2-4.

Data Input
Data to be written into the scratchpad is channeled from the SSMUX and clocked into the scratchpad registers.

Addressing the Scratchpad
The address of the scratchpad memory register to be accessed is generated by the scratchpad address multiplexer (SPAM), located on the control module (K2-4). Depending on the state of the select lines to the SPAM, the source of the address can be any of the following:

1. The Control Store ROM (ROMSPA03:ROMSPA00).
2. Instruction Register Source Field (IR08:IR06)
3. Instruction Register Destination Field (IR02:IR00)
4. Bus Address (PBA03:00)

Reading from the Scratchpad

If the Control Store circuitry forces a low on the K1-10 ENAB GR L line at the beginning of a machine cycle, the tristate outputs of the scratchpad will be enabled. Ninety or 120 nanoseconds after the cycle begins (allows the scratchpad address to set up), K1-5 TAP 30 H goes low, allowing data stored in the selected scratchpad register to be latched in the output buffer SP15:SP00 lines. This data will continue to be read during the rest of the machine cycle. (See Figure 4-4.) Table 4-3 shows the various scratchpad enabling configurations and the modes they select.

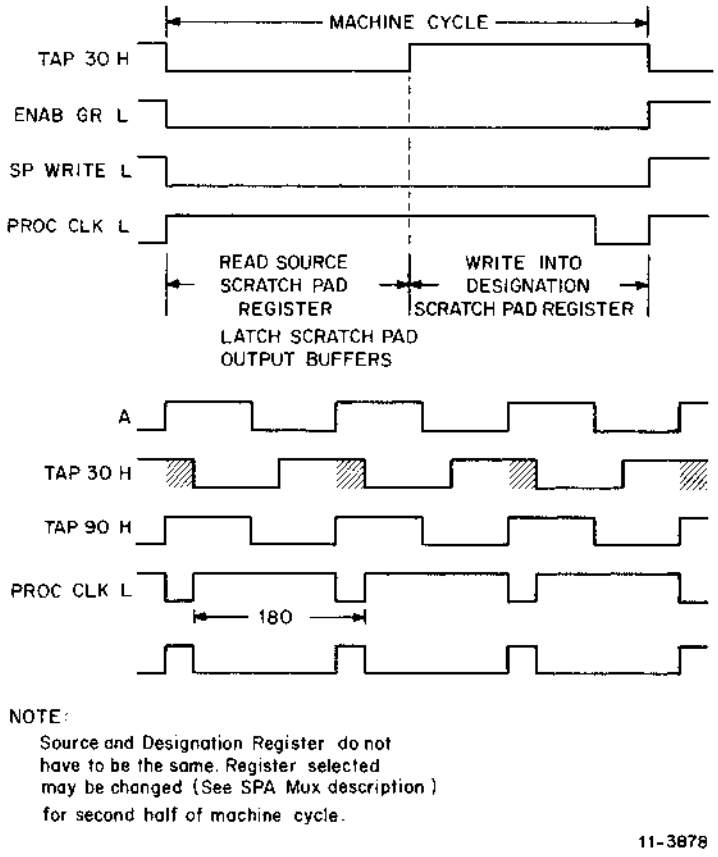


Figure 4-4 Scratchpad Timing

Table 4-3 Scratchpad Enabling Configurations and Modes

OD	WE	CLK	OS	Mode	Outputs
L	X	X	L	OUTPUT STORE	Data from last addressed location
X	L	⌐	X	WRITE DATA	Data being written (if OD = L and OS = H)
L	X	X	H	READ DATA	Data stored in addressed location
H	X	X	L	OUTPUT STORE	High-impedance state
H	X	X	H	OUTPUT DISABLE	High-impedance state

Latching of Outputs
When the OD (pin 12) and OS (pin 13) inputs are both low, the data being read from the scratchpad that is addressed is latched into the buffers on the output of scratchpad memory (SP15–00). Once those outputs are stabilized, they are not affected by any modifications to the scratchpad memory address lines for the remainder of the cycle.

Clocking the Scratchpad
The REG CLK H clock signal clocks data from the SSMUX lines into the scratchpad register and writes that data into scratchpad memory. TAP 30 H unasserted, placing a high at the OS input (pin 13) of the scratchpad, is all that is required for a read operation. Both a read and a write can take place during the same machine cycle. Figure 4-4 shows the scratchpad timing for one machine cycle.

Scratchpad Address Multiplexer (SPAM)
The SPAM (Figure 4-5) generates the four address signals that select the desired scratchpad register, or word. The SPAM (shown in print K2-4 of D-CS-M7266-0-1) consists of two 74S153 dual 4-line-to-1-line data multiplexers, or a total of four 4-to-1 multiplexers, all with a common strobe input signal (GND) and common address input signals (S1 and S0). Four data input sources are connected so that, when the SPAM is addressed and strobed, it generates one 4-bit output, selected from one of the four sources. Table 4-4 lists the sources of SPAM input data and the address input signal configurations that select them.

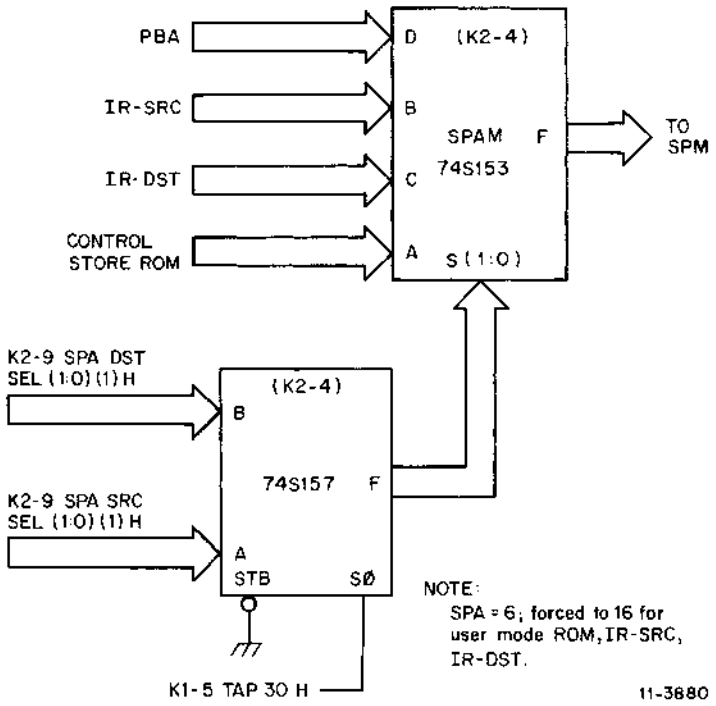


Figure 4-5 Scratchpad Address Multiplexer (SPAM)

Table 4-4 SPAM Input Data Sources

Function	SPAM Input	Source	Source Print	S1	S0
Source Operand Register Selection	B	Instruction Register Bits 08:06	K2-5	L	H
Destination Operand Register Selection	C	Instruction Register Bits 02:00	K2-5	H	L
General-Purpose Register Selection from Console	A	ROM SPA Bits 03:00	K2-10	L	L
Register Selection by Microprogram	D	PBA Bits 03:00	K1-6	H	H

Scratchpad Memory Organization

The scratchpad memory (SPM) is a 16-word-by-16-bit random access read/write memory composed of four 16-word-by-4-bit bipolar (85S68) memory units (K1-1 through K1-4). The 16-word-by-16-bit organization of this memory provides 16 storage registers that are utilized as shown in Table 4-5.

Table 4-5 SPM Register Utilization

Register Number	Description
R0, R1, R2, R3, R4, R5	General-Purpose Registers
R6	(Processor Stack Pointer)
R7	(Program Counter)
R10	Temporary Storage
R11	Unused
R12	Temporary Storage
R13	Temporary Storage
R14	Unused
R15	Temporary Storage
R16	Processor Stack Pointer (Memory Management User Mode)
R17	Temporary Storage

Scratchpad Outputs
Data outputs from the scratchpad are fed to the ALU as the A leg input and to the memory management system.

4.2.4 B Leg

The B leg (Figure 4-6) of the ALU consists of three components: the B register, the BX register, and the B leg multiplexer (BMUX). Each of these components is divided into four 4-bit slices, with circuit schematic prints K1-1, K1-2, K1-3, and K1-4 each containing a slice. Data from the SSMUX can be clocked into either register. Register contents can be shifted either individually as 16-bit words or together as a double (32-bit) word.

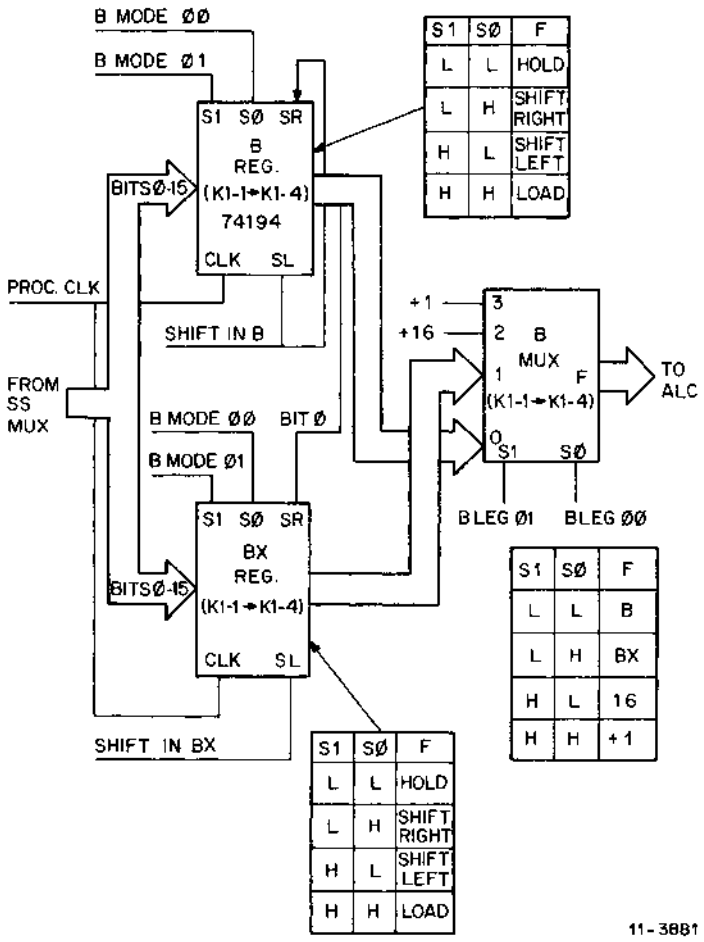


Figure 4-6 B Leg Block Diagram

B Register

The B register (B REG) is a general-purpose storage register (Figure 4-7) on the B leg of the ALU, consisting of four 4-bit bidirectional universal shift registers (74194). The mode control lines of the four 4-bit registers are connected in parallel, so that the signals K2-8 B MODE 00 L and K2-8 B MODE 01 L select the function that will be performed by the B register when clocked by K1-5 PROC CLK L. Table 4-6 shows the various functions and the shift configurations that select them.

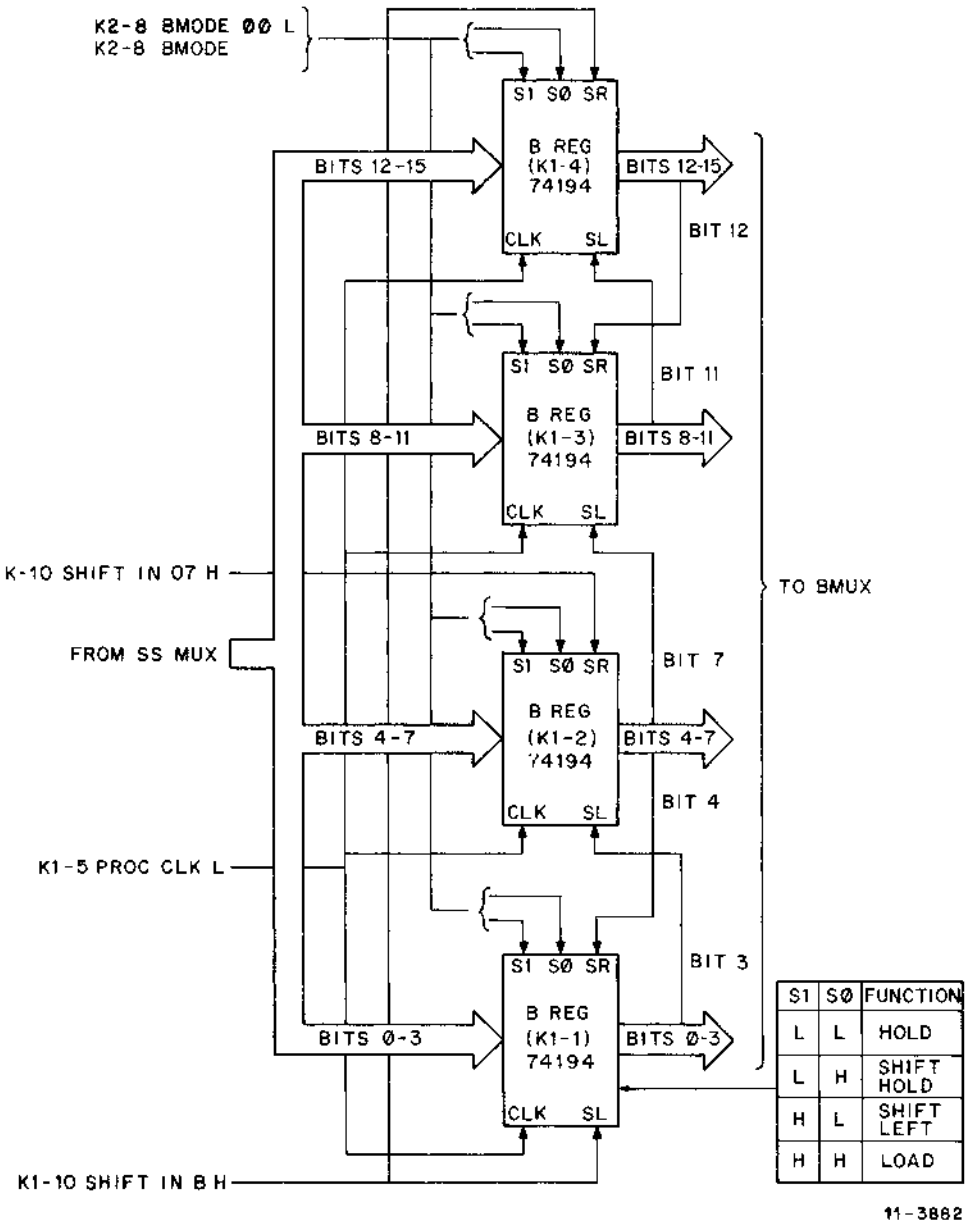


Figure 4-7 BREG Block Diagram

Table 4-6 B and BX Register Enabling Configurations and Modes

Mode 01	Mode 00		Function (when PROC CLK L goes high)
L	L	Hold	Contents of register do not change.
L	H	Shift Right	Contents are shifted right one bit.
H	L	Shift Left	Contents are shifted left one bit.
H	H	Parallel Load	Data from SSMUX is loaded into B register and appears at output.

The B register can be shifted as an 8-bit byte or a 16-bit word. The signal K1-10 SHIFT IN B determines what is shifted into the B register. When the contents of this register and the BX register are combined into a 32-bit word, the B register contains the upper 16 bits.

BX Register

The BX register (BX REG) is a general-purpose storage register (Figure 4-8) on the B leg of the ALU, consisting of four 4-bit bidirectional universal shift registers (74194), similar to the B register. The mode control lines of the four 4-bit registers are connected in parallel, so that the signals K2-8 BX MODE 00 L and K2-8 BX MODE 01 L select the function to be performed when the BX REG is clocked by K1-5 PROC CLK L. The BX register can be shifted as a 16-bit word or, in conjunction with the B register, as a 32-bit word. In the latter case, the BX register contains the lower 16 bits of the 32-bit word, and the shift right (SR) input of the most significant register in the BX register is connected to the zero bit of the B register. Table 4-6 shows the various functions and the shift configurations of K2-8 BX MODE 00 L and K2-8 BX MODE 01 L that select them.

B Leg Multiplexer (BMUX)

The BMUX (Figure 4-9) consists of three 2-to-11 multiplexers and a 4-to-1 multiplexer, and is used to select the proper input to be used as an operand on the B leg of the ALU. The BMUX can select the contents of either the B REG or BX REG, or can act as a constant generator (constants 16, 1, or 0), depending on the configuration of signals K2-8 B LEG 00 H and B LEG 01 H (Table 4-7) and the state of K2-4 DISAB MSYN +1 L.

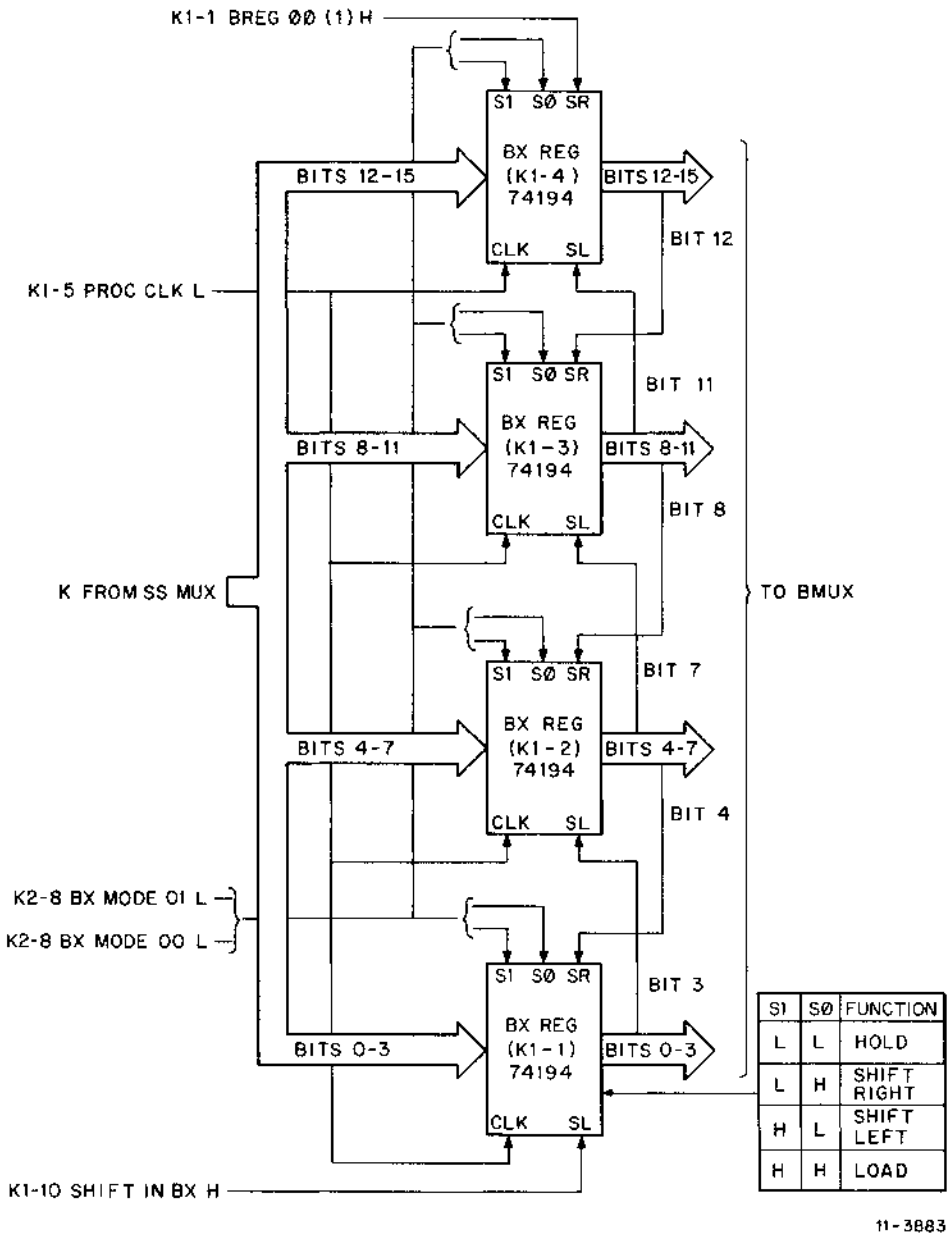


Figure 4-8 BX REG Block Diagram

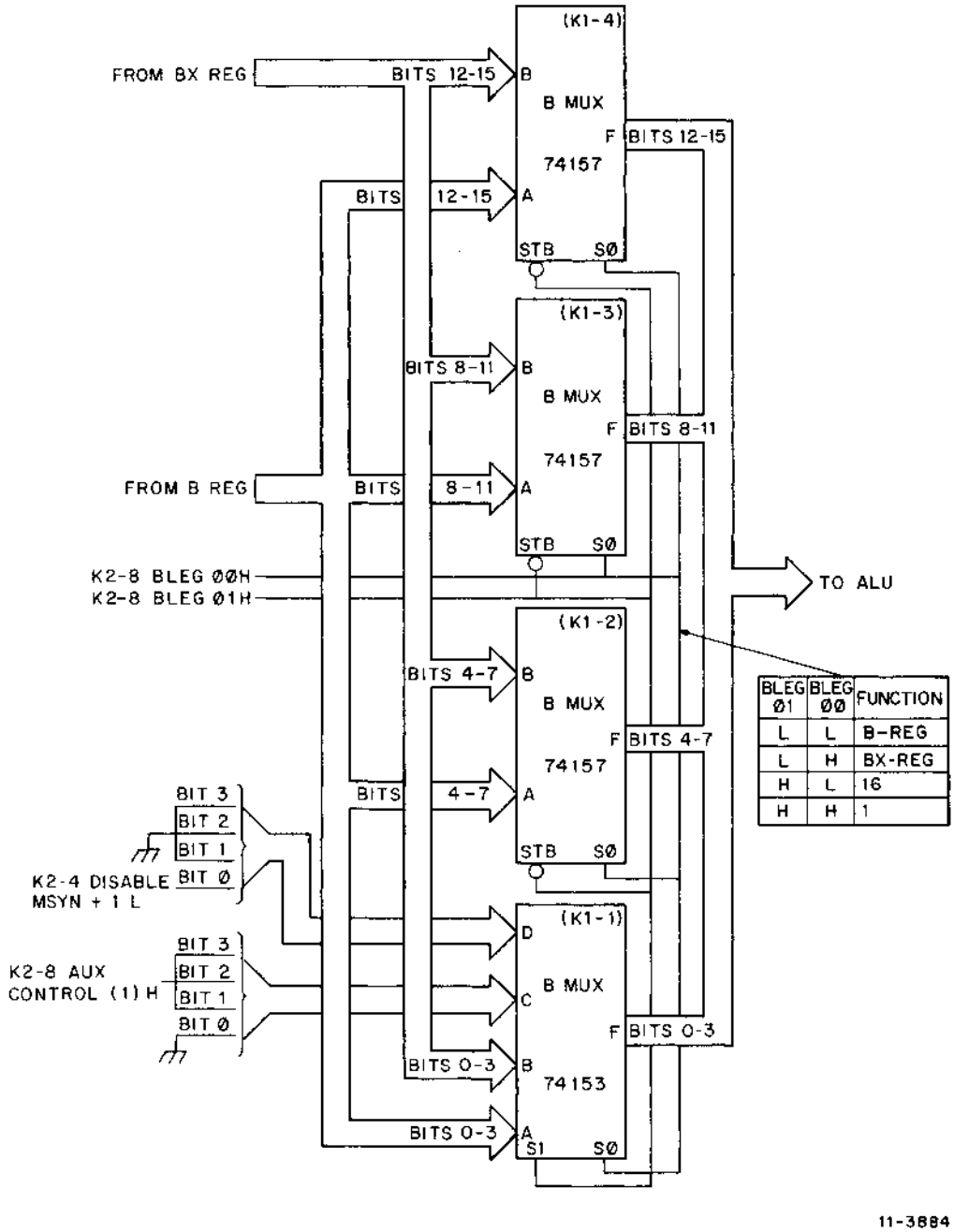


Figure 4-9 BMUX Block Diagram

Table 4-7 BMUX Enabling Configurations and Modes

B Leg 01	B Leg 00	Function	Description
L	L	B REG	Passes data from the B register to the BMUX outputs. This is the most common configuration.
L	H	BX REG	Passes data from the BX register to the BMUX outputs. This is used principally for EIS instructions.
H	L	+16	Forces the constant +16 into the BMUX outputs to preset a counter that is used for EIS instructions.
H	H	+1	Forces the constant +1 into the BMUX outputs during operations in which the contents of a register are being incremented or decremented by two.
-	-	0	By asserting DISABLE MSYN +1 L, this configuration forces the constant 0 into the BMUX outputs during operations in which the contents of a register are being incremented or decremented by one. (The signal K2-8 ALU CIN L to the ALU from the control module provides the one.)

B Leg Shift Capabilities

Each of the four shift registers (74194) that make up each register (B REG and BX REG) has the capability of being shifted left or right, as indicated in Table 4-6 and Figure 4-10. The B register can be shifted as an 8-bit byte or a 16-bit word; the BX register can be shifted as a 16-bit word or, in conjunction with the B register, as a 32-bit word.

Byte Shifts

If the mode control lines (K2-8 B MODE 00 L and K2-8 B MODE 01 L) specify a shift left, B REG 15:00 are shifted one position toward the most significant bit at the clock pulse K1-5 PROC CLK L going high. The signal K1-10 SHIFT IN B H is shifted into bit 00 via the SL input. This signal is generated by the SHIFT MUX (E117 on print K1-10) as a function of the select signals K2-8 SHIFT MUX 01 L and K2-8 SHIFT MUX 00 L. The shift right input to B REG bits 07:04 comes from the BYTE MUX (E106 on print K1-10). Assertion of K2-5 BYTE L (indicating a byte instruction) causes bit 07 of the B REG to be loaded directly by K2-5 SERIAL SHIFT H; if K2-5 BYTE L is high, however, B REG bit 07 is loaded from B REG 08. B REG bit 15 is loaded from K1-10 SHIFT IN B H during a shift right (just as B REG bit 00 is loaded during a shift left), and can be loaded with itself, K2-5 SERIAL SHIFT H, ground, or BX REG bit 15, depending on the SHIFT MUX. For a shift right, BX REG bits 15:01 are shifted one position toward the least significant bit, and BX REG bit 15 is loaded with B REG bit 00. Thus, for all right shifts, the BX REG acts as the low-order 16 bits of a 32-bit word made up of B REG and BX REG. For a shift left, BX REG bits 15:00 are shifted one bit position toward the most significant bit. BX REG bit 00 is loaded with the signal K1-10 SHIFT IN BX H, which is generated by the SHIFT MUX. Depending on the configuration of the SHIFT MUX control lines K2-8 SHIFT MUX 00 L and K2-8 SHIFT MUX 01 L, the BX REG may be loaded with any of four possible inputs: K1-4 ALU COUT H, the output of the EIS overflow detection logic (E98 on print K1-10), ONE, and ZERO.

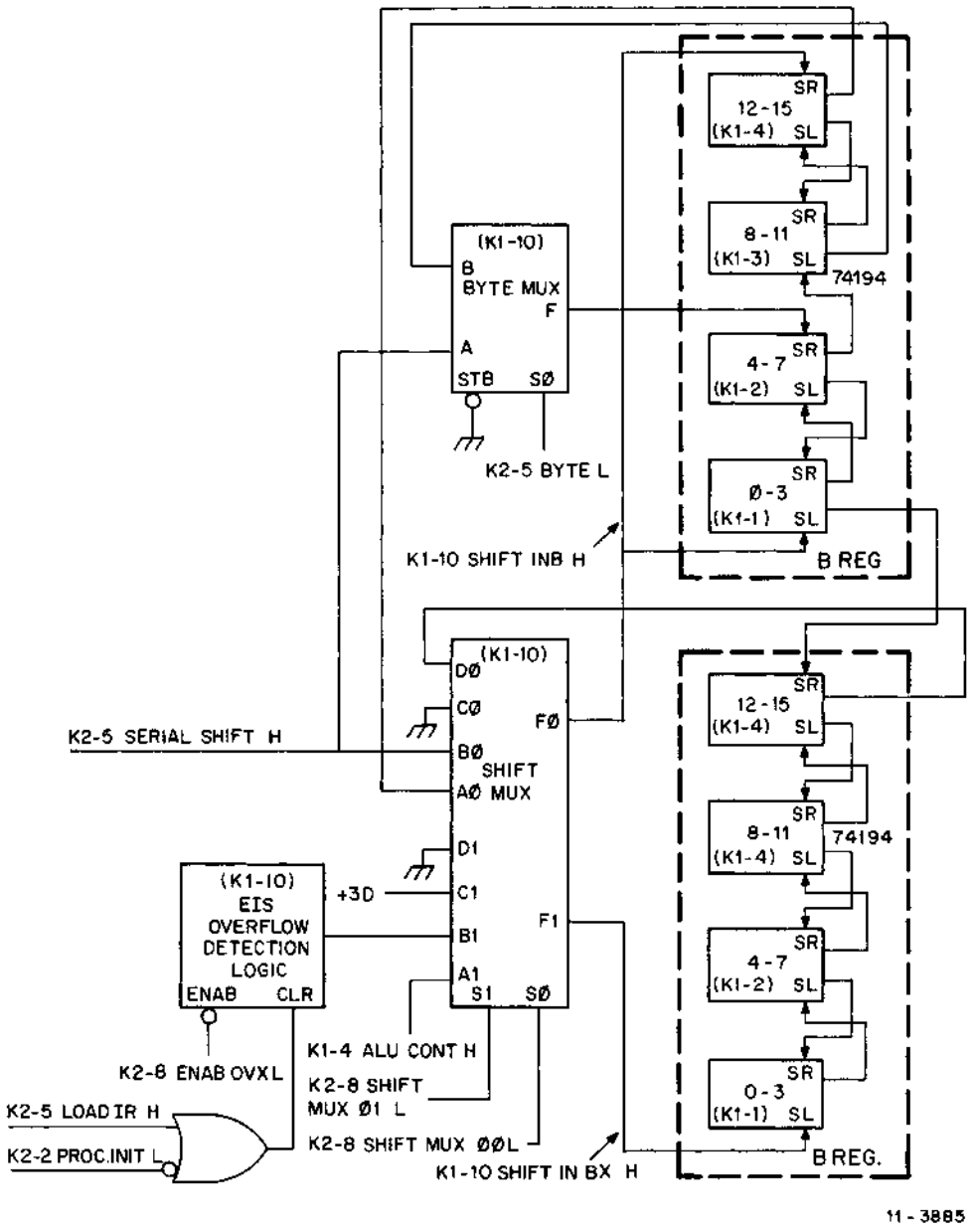


Figure 4-10 B Leg Shift Logic

Specific Shift and Rotate Operations
The shifting requirements for the ASL, ASR, ROL, ROR, ASH, and ASHC instructions are described briefly below.

1. Arithmetic Shift Left (ASL) – Shifts all bits of the destination left one place. The low-order bit is loaded with a 0. The C-bit of the status word is loaded from the high-order bit of the destination. ASL performs a signed multiplication of the destination by 2, with overflow indication.

2. Arithmetic Shift Right (ASR) – Shifts all bits of the destination right one place. The high-order bit is duplicated. The C-bit is loaded from the low-order bit of the destination. ASR performs signed division of the destination by two.

3. Rotate Left (ROL or ROLB, depending on whether a word or byte operation) – Rotates all bits of the destination left one place. The high-order bit is loaded into the C-bit of the status word, and the previous contents of the C-bit are loaded into the low-order bit of the destination.

4. Rotate Right (ROR or RORB) – Rotates all bits of the destination right one place. The low-order bit is loaded into the C-bit, and the previous contents of the C-bit are loaded into the high-order bit of the destination.

5. Arithmetic Shift (ASH) – Shifts the contents of the register right or left the number of times specified by the source operand. The shift count is taken as the low-order six bits of the source operand. This number ranges from –32 to +31. Negative is a right shift and positive is a left shift.

6. Arithmetic Shift Combined (ASHC) – Treats the contents of the register and the register ORed with one as one 32-bit word. Rv1 (bits 15:00) and R (bits 31:16) are shifted right or left the number of times specified by the shift count. The shift count is taken as the low-order six bits of the source operand. This number ranges from –32 to +31. Negative is a right shift and positive is a left shift. (When the register chosen is an odd number, the register and the register ORed with one are the same. In this case, the right shift becomes a rotate. The 16-bit word is rotated right the number of bits specified by the shift count.)

NOTE

When R is an even-numbered register, Rv1 will be the next highest register. If R is an odd-numbered register, Rv1 will be the same register (e.g., if R = R4, then Rv1 = R5; if R = R5, then Rv1 = R5).

BMUX Operation

Three 2-to-1 multiplexers (74157s) are used to switch B leg bits 15:04. Their select lines are tied in parallel with each other and with the S0 line of the 4-to-1 multiplexer (two 74153s) used to switch B leg bits 03:00. The S0 line is signal K2-8 B LEG 00 H. Signal K2-8 B LEG 01 H is connected to the enable lines of the 2-to-1 multiplexers and to the S1 line of the 4-to-1 multiplexer. Table 4-7 describes the enabling configurations and modes for these two select signals, which are logically determined as follows:

1. If both K2-8 B LEG 00 H and K2-8 B LEG 01 H are low, the 4-to-1 multiplexer (E8 and E5 on print K1-1) selects the A input and the 2-to-1 multiplexers (E28 on print K1-2, E18 on K1-3, and E38 on K1-4) select the A inputs; the data from the B REG is switched to the BMUX output.

2. If K2-8 B LEG 01 H is low and K2-8 B LEG 00 H is high, the 4-to-1 multiplexer selects input B, the 2-to-1 multiplexers remain enabled, and the data from the BX REG is switched to the BMUX output.

3. If K2-8 B LEG 01 H is high and K2-8 B LEG 00 H is low, the 2-to-1 multiplexers are not enabled. The 4-to-1 multiplexer selects input C, where bit 0 is low and bits 03:01 are connected to K2-8 AUX CONTROL (1) L unasserted, generating a +16 constant to the B leg.

4. If both K2-8 B LEG 01 H and K2-8 B LEG 00 H are high, the 2-to-1 multiplexers are still disabled and the 4-to-1 multiplexer selects input D, where bits 03:01 are grounded and bit 00 is connected to K2-4 DISABLE MSYN +1 L unasserted, generating a constant of +1 to the B leg.

5. If, in 4 above, K2-4 DISABLE MSYN +1 L is asserted, a constant of 0 is generated to the B leg.

Constants +16, +1, and 0

The purpose of generating the constants +1 and 0 on the B leg input of the ALU is to aid the processor to perform autoincrement and autodecrement operations. During either operation, if a word instruction is being performed, the specified register is incremented or decremented by two; if a byte instruction is being performed, the register is incremented or decremented only by one. The actual ALU operation is:

RESULT = A LEG DATA + B LEG DATA + ALU CIN.

The ALU always uses the K2-8 ALU CIN L signal to increment or decrement the A leg input by one; thus, the B leg input must provide the constant +1 or 0 to obtain the correct autoincrement or autodecrement result for both byte and word instructions. A B leg constant of +1 is generated by enabling the least significant bit of the BMUX output (bit 00) and forcing all other bits (15:01) to 0. To generate a constant 0, even bit 00 is cleared. The actual constant generated is defined by the state of the K2-4 DISABLE MSYN +1 L signal, which is determined by the Control Store.

4.2.5 ALU Multiplexer (AMUX)

The AMUX (Figure 4-11) consists of three 4-to-1 multiplexers (74S153s) and one 2-to-1 multiplexer (74S157), each one dedicated to a 4-bit slice of the AMUX. The 2-to-1 multiplexer (E14 on print K1-3) switches AMUX bits 11:08 according to the state of the STB and S0 inputs. If the STB input is high, the multiplexer is disabled, and the output is forced low. If the STB input is low, the multiplexer is enabled, and the output depends on the state of the S0 input and the appropriate data input. Thus, if the S0 input is low, the A data input will be gated through to the AMUX output; if the S0 input is high, the B data input will be gated through. Because the 4-to-1 multiplexer does not have an enable input, the output always follows one of the inputs, corresponding to the binary value of select lines S1 and S0 (K1-10 AMUX S1 H and K1-10 AMUX S0 H, respectively), as follows:

1. Unibus Data Function – If both S1 and S0 are low (binary 0), the 4-to-1 multiplexers select input A and the 2-to-1 multiplexer selects input A. Thus, each 4-bit slice of the AMUX switches Unibus data into the data path.

2. Constant's Function – Certain operations require the introduction of specific numbers into the data path. (For example, the data path must generate a vector of 24 for a power-fail trap, or 114 for a parity trap.) Access to these and other numbers is facilitated by storing certain constants in a read-only memory and presenting them to the constant's input of the AMUX. If S1 is low and S0 is high (binary 1), the 4-to-1 multiplexers select input B (the constant's input). Bits 11:08, which are controlled by the 2-to-1 multiplexer, are not used.

3. ALU Input – If S1 is high and S0 is low (binary 2), the 4-to-1 multiplexers select ALU inputs (input C) and the 2-to-1 multiplexer is enabled, also selecting ALU inputs, so that the ALU lines are selected for all 16 bits.

4. PSW Inputs – If both S1 and S0 are high (binary 3), the 4-to-1 multiplexers select the PSW input (input D). The 2-to-1 multiplexer is disabled, as bits 11:08 are not used.

4.2.6 Processor Status Word

The Processor Status Word (PSW) register contains information on the current and previous memory management mode, the current processor priority, a processor trap for debugging, and the condition code results of the previous operation. The PSW bit assignments and uses are shown in Table 4-8.

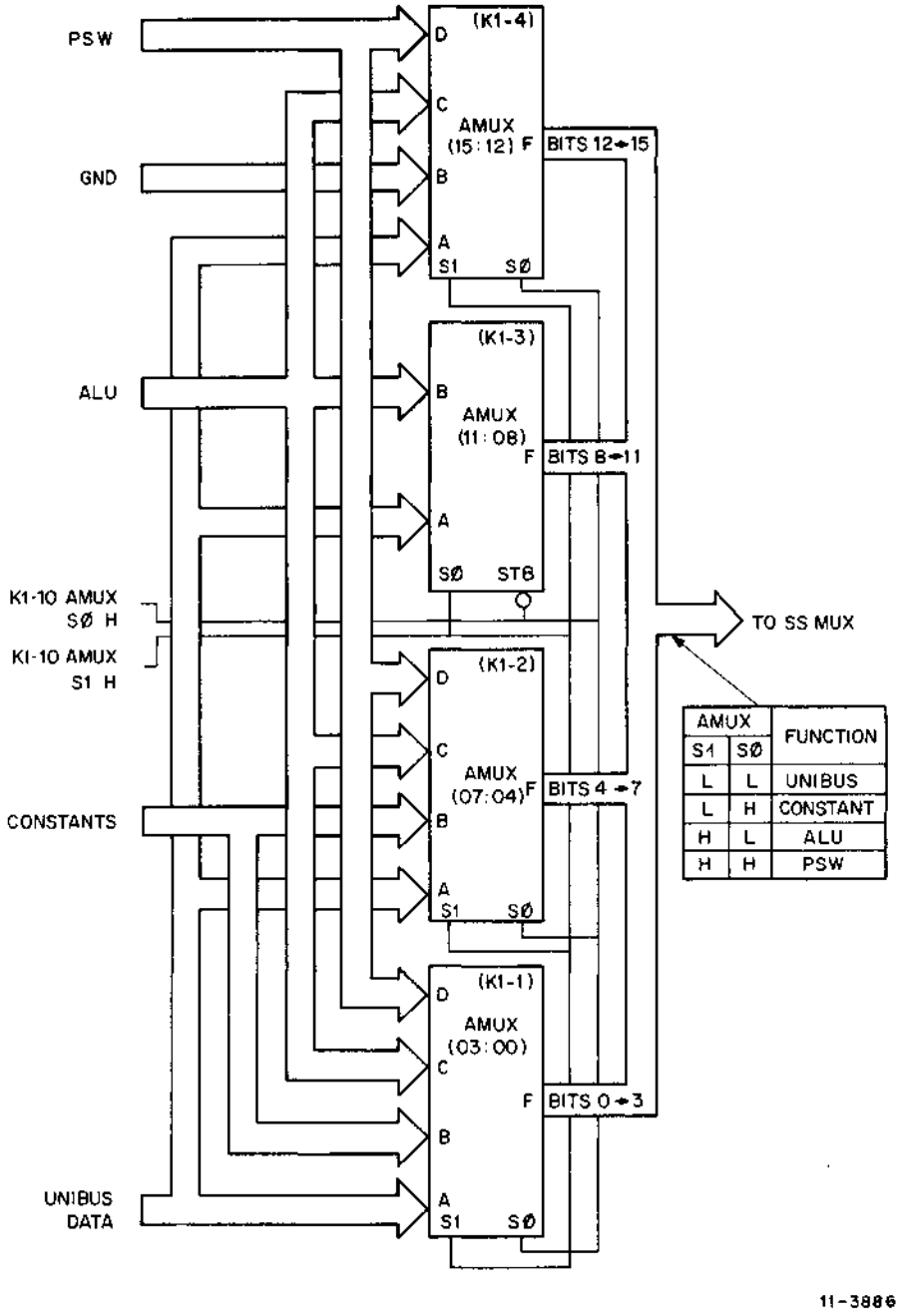


Figure 4-11 AMUX Block Diagram

Table 4-8 Processor Status Word Register Bit Assignments

PSW Bit	Name	Use
15:14	Memory Management Current Mode	Contain the current memory management modes.
13:12	Memory Management Previous Mode	Contain the previous memory management modes.
11:08	Unused	
07:05	Priority	Set the processor priority.
04	Trace	When this bit is set, the processor traps to the trace vector. Used for program debugging.
03	N	Set when the result of the last data manipulation is negative.
02	Z	Set when the result of the last data manipulation is zero.
01	V	Set when the result of the last data manipulation produces an overflow.
00	C	Set when the result of the last data manipulation produces a carry from the most significant bit.

The PSW (Figure 4-12) is a 12-bit register composed of three quad D-type flip-flops (74175s) and one separate D-type latch. The first of these (E95 on print K1-1) stores the condition code bits (N, Z, V, and C), and derives its input from the PSW MUX, a quad 2-line-to-1-line multiplexer (E96 on K1-1) according to the state of the S0 select line. When high, S0 selects the B inputs (SSMUX bits 03:00); when low, S0 selects the A inputs, which come from the condition code logic (print K1-10). The selected inputs are passed to the f-outputs of the multiplexer and into the PSW.

A second quad D-type flip-flop (E97 on K1-2) is used to store the three KD11-E processor priority bits, which it obtains from SSMUX bits 07:05. A separate 74S74 (E107 on K1-2) is needed to store the Trace Trap flag (T-bit), which can be loaded from the K1-2 SSMUX 04 H line.

The third quad D-type flip-flop (E80 on K1-4) stores the bits containing the current and previous status of the memory management mode. SSMUX bits 15 and 14 provide the input for PSW bits 15 and 14, which are then rerouted through a quad 2-line-to-1-line multiplexer (E90 on K1-4) and multiplexed with SSMUX bits 13 and 12 according to the state of the S0 select signal [K2-9 FORCE KERNEL (1) H] to provide the input for PSW bits 13 and 12. Thus, PSW bits 15 and 14 reflect the current status of the memory management mode, while PSW bits 13 and 12 indicate the previous status.

All flip-flops in the PSW are clocked, directly or indirectly, by clocking signal K1-5 REG CLK L. All of the enabling signals come from the Control Store.

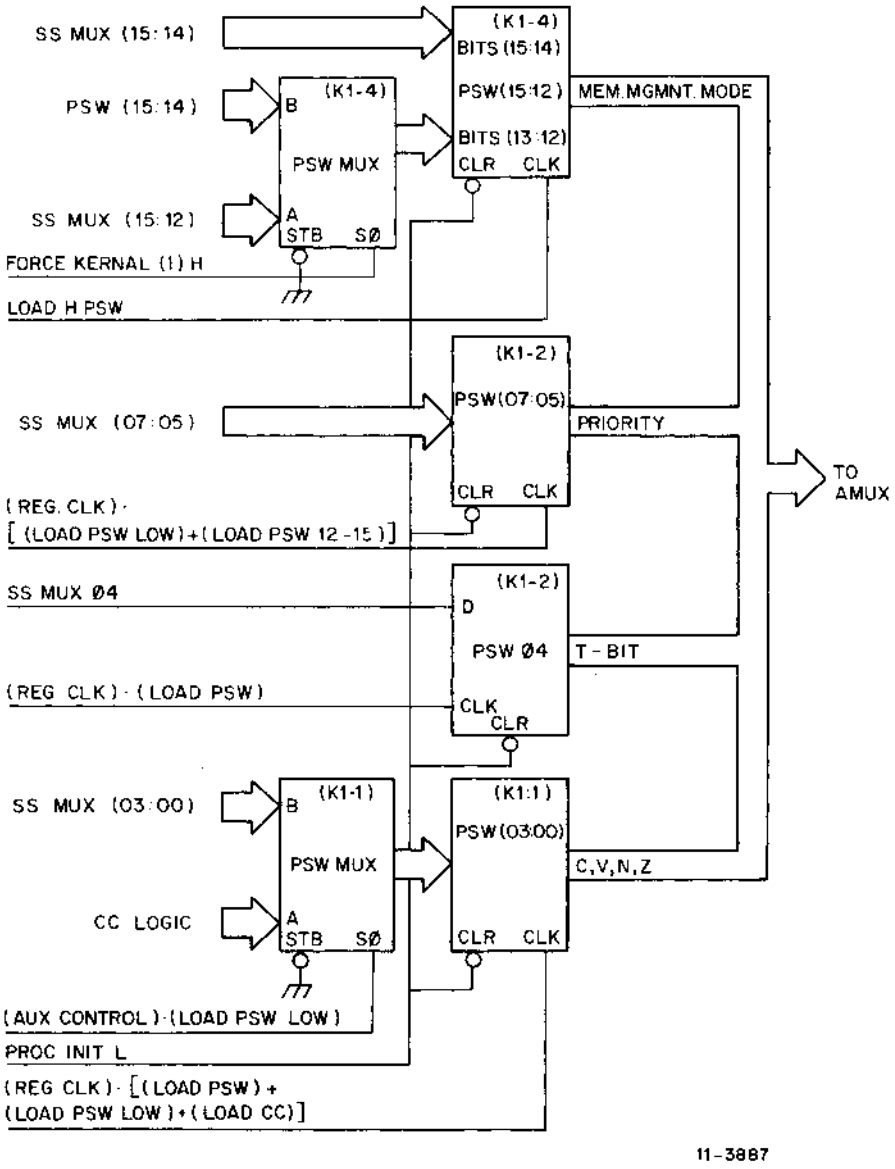


Figure 4-12 Processor Status Word

4.3 CONDITION CODES

The logic necessary for determining the condition codes is shown on sheets K1-10 and K2-5, and can be subdivided into three parts, each of which is discussed in some detail in this section. Constraints for each condition code bit are shown in the instruction set specifications (Chapter 2).

4.3.1 Instruction Categorizing ROM

The Categorizing ROM (E67 on sheet K2-5) decodes the instructions in the IR and categorizes them into eight groups, based on their effect on the carry and overflow condition codes. These groups are as follows:

Group	Instructions
1	MOV, BIT, BIS, BIC, and non-PDP-11 instructions
2	INC, DEC
3	CLR, TST, SWAB
4	ADD, ADC
5	NEG, CMP, COM
6	SUB, SBC
7	Rotate instructions
8	Unused

Three of the four outputs of the Categorizing ROM are used to provide a binary representation of one of the above instruction categories for the C and V Decode ROM (E105 on K1-10). The fourth output (K2-5 BYTE L) decodes the fact that the instruction in the IR is a byte instruction and is fed to the select input of the BYTE MUX (E106 on K1-10).

4.3.2 Byte Multiplexer (BYTE MUX)

The BYTE MUX (E106 on K1-10) is a quad 2-line-to-1-line multiplexer (74S157) that determines the N condition code bit and the K1-10 SHIFT IN 07 H signal for the B REG (Figure 4-13). A single select input (K2-5 BYTE L) selects the A inputs when a byte operation is performed, and the B inputs when the operation is not a byte.

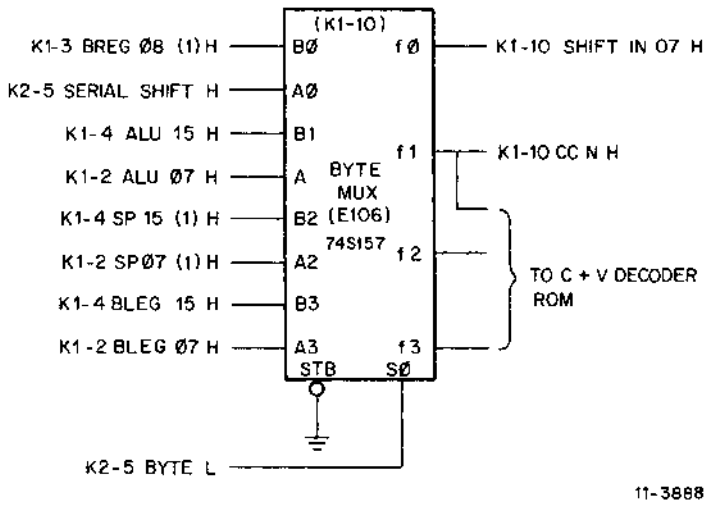


Figure 4-13 Byte Multiplexer

Output signal K1-10 CC N H assumes the level of K1-4 ALU 15 H when the instruction being performed is a word operation, and the level of K1-2 ALU 07 H when the instruction is a byte operation. Byte operations may be performed on either the high or low bytes of the input word, depending on whether the processor microcode has already swapped bytes before the condition codes are detected.

For shift right operations, the K1-10 SHIFT IN 07 H output assumes the level of the K1-3 BREG 08 (1) H input when a word instruction is performed, and the level of the K2-5 SERIAL SHIFT H output of the ROT/SHFT ROM (E61 on print K2-5) for a byte operation. The diagrams in Figure 4-14 indicate the operations performed by various instructions.

4.3.3 C and V Decode ROM

The C and V Decode ROM (E105 on K1-10) determines the values of the carry and overflow condition code bits as a function of the instruction being performed (Figure 4-15). Inputs to this ROM come from the ROT SHIFT ROM (E61 on K2-5), the PSW [K1-1 CBIT (1) H], the BYTE MUX, and the Categorizing ROM (E67 on K2-5). Outputs K1-10 CC V H and K1-10 CC C H are fed via the PSW MUX (E96 on K1-1) to the PSW register.

4.3.4 Condition Code Signal CC Z H

Each 4-bit slice of the data path contains an ALU output via a gate (type 8815) reflecting whether all four of the bits in that slice are ZERO. If the instruction being performed is a byte operation, condition code signal K1-10 CC Z H assumes the combined state of signals K1-1 0–3=0 H and K1-2 4–7=0 H; for a word operation, K1-10 CC Z H assumes the combined state of those signals together with K1-3 8–11=0 H and K1-4 12–15=0 H. Thus, K1-10 CC Z H is asserted if bits 00 through 07 = 0 for a byte operation and if bits 00 through 15 = 0 for a word operation. Assertion of K2-5 BYTE L selects byte operation.

4.4 UNIBUS ADDRESS AND DATA INTERFACE

4.4.1 Unibus Drivers and Receivers

Standard bus transceiver circuits (type 8641) are used to interface the processor data path to the Unibus address (BUS A00:A15) and data (BUS D00:D15) lines. These circuits are shown on prints K1-1 through K1-4, and on K1-6. Figure 4-16 shows the logic diagram for an 8641.

4.4.2 Unibus Address Generation Circuitry

A unique feature of the KD11-E is that KT11-D equivalent memory management capability is built into the 2-board processor. During Unibus transfers, virtual bus addresses are obtained from the scratchpad memory (SPM) and the Physical Bus Address (PBA) register, if relocation is not enabled, and latched in the Virtual Bus Address (VBA) register shown on print K1-6. Figure 4-17 shows the actual VBA clock timing, while Figure 4-18 shows Unibus address logic in block diagram form.

If the memory management circuit is not enabled (K1-8 RELOCATE H is not asserted), the address that was clocked into the Physical Bus Address register is used as address data for the 8641 transceivers and driven onto the Unibus address lines.

When the memory management circuit is enabled (K1-8 RELOCATE H asserted), a selected relocation constant (detailed description in Paragraph 4.12) is added to the contents of the VBA before it is latched into the BA and driven onto the Unibus.

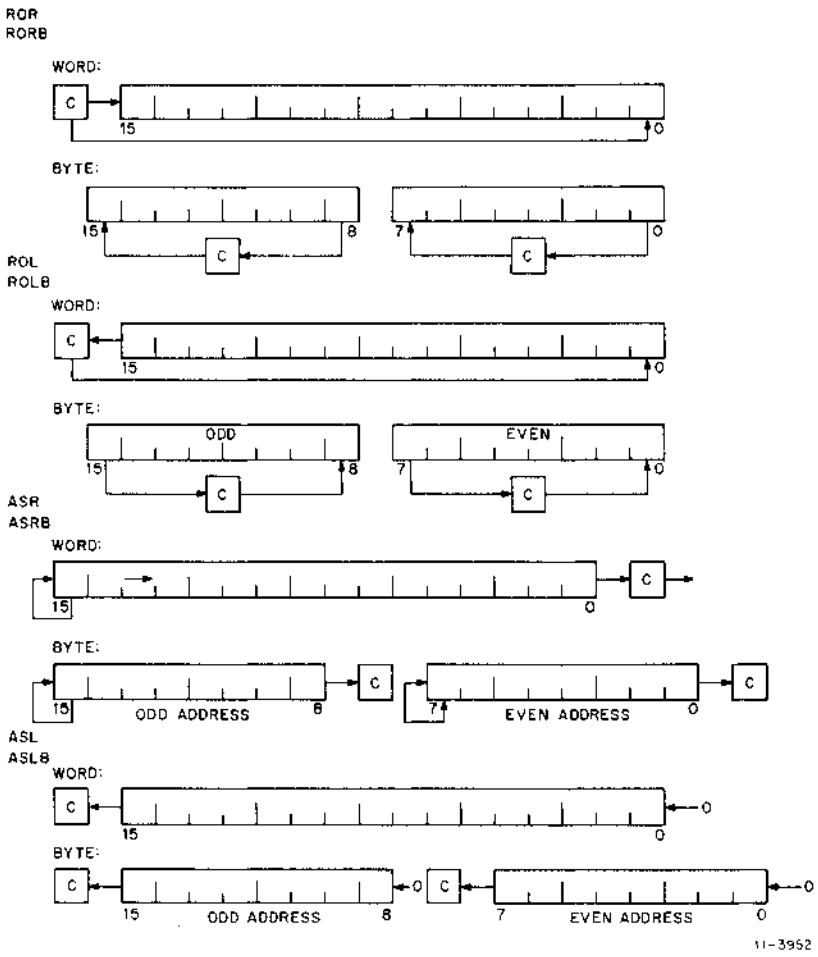


Figure 4-14 Rotate Instructions

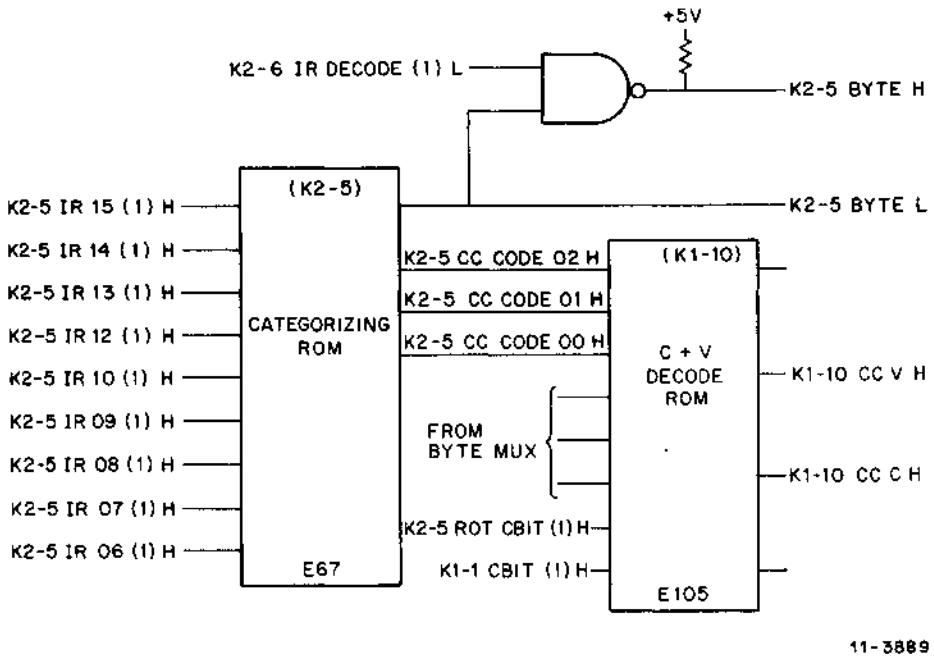


Figure 4-15 C and R Decode ROM

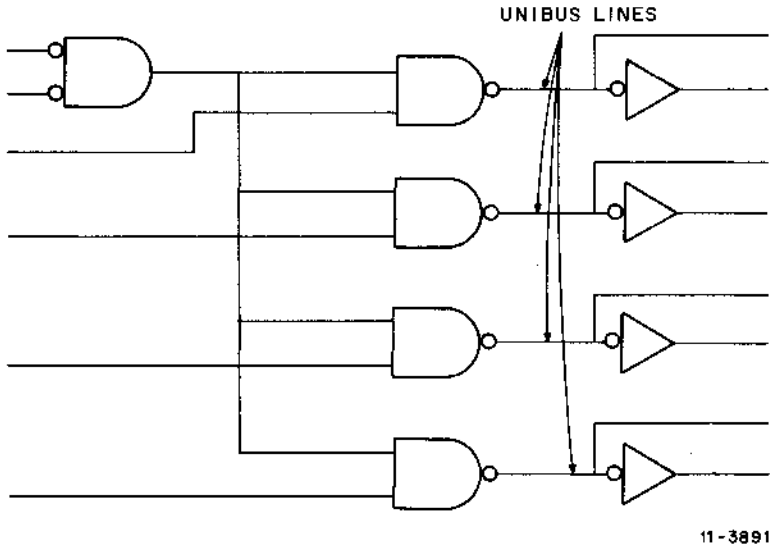


Figure 4-16 Unibus Transceiver

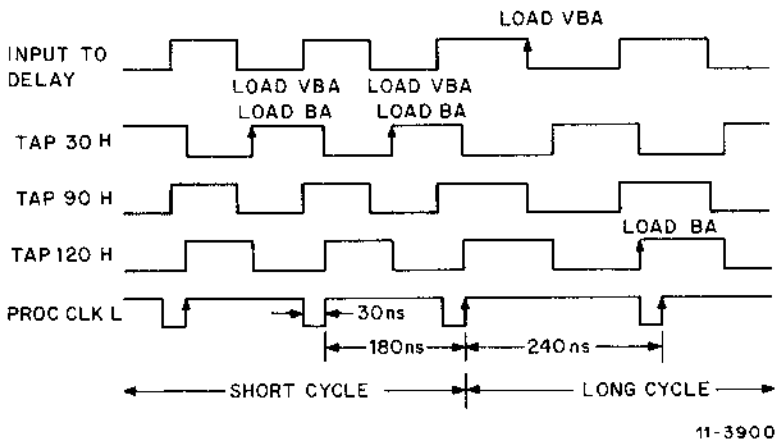


Figure 4-17 Processor Clock Cycle Timing

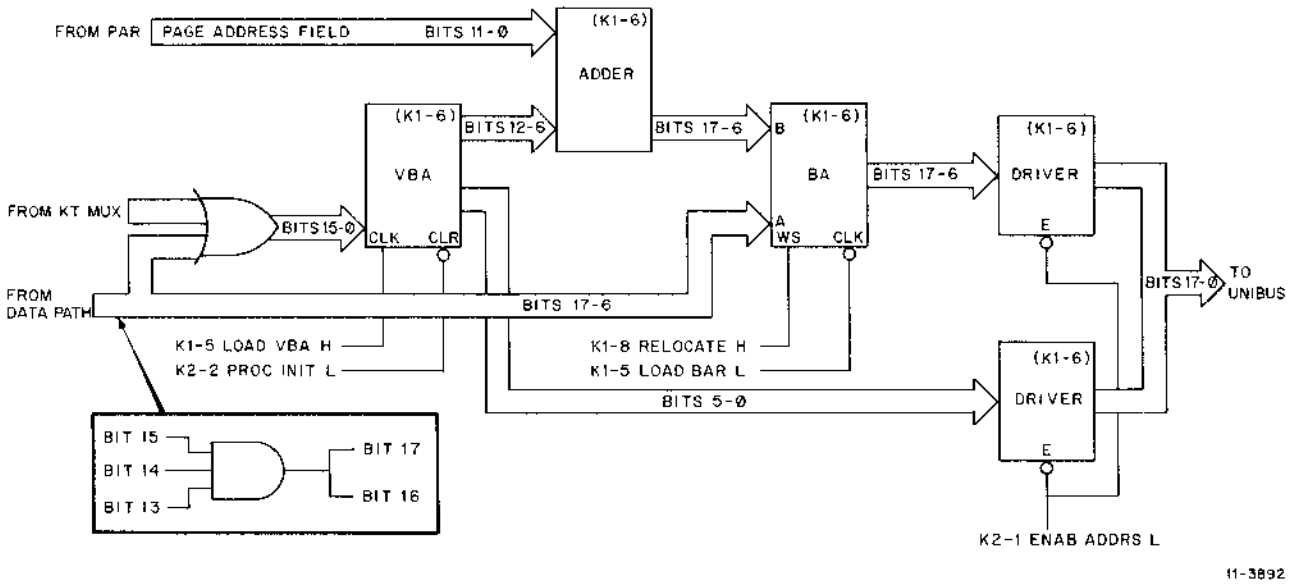


Figure 4-18 Unibus Address Logic Block Diagram

4.4.3 Internal Address Decoder

The receiver half of the bus transceivers continually monitors the Unibus address lines. If the processor is *running* (HALT RQST L or BUS SACK L are not asserted), these transceivers allow the Internal Address Decoder circuit (print K1-10) to detect transfers to or from the PSW and memory management registers. Note, however, that the CPU does not allow access to its general registers through their Unibus addresses while it is running.

While the processor is halted (BUS SACK L is asserted), this decoder circuit enables data transfers between CPU registers and Unibus peripheral devices. A list of these CPU registers and their Unibus addresses is shown below; the registers are discussed in Paragraph 4.12.

PSW	777776	R10	777710
R0	777700	R11	777711
R1	777701	R12	777712
R2	777702	R13	777713
R3	777703	R14	777714
R4	777704	R15	777715
R5	777705	R16	777716
R6	777706	R17	777717
R7	777707		

4.5 INSTRUCTION DECODING

4.5.1 General Description

Two methods are used to control instruction decoding, one using microroutine selection and the other using auxiliary ALU control. Dual control is required because of the large number of instructions that require source/destination calculations. Auxiliary ALU control is evoked whenever the microcode executes the action X = Y OP B as a result of a specific instruction.

There are two prerequisites to a thorough understanding of the instruction decoding procedure. One is a knowledge of the microbranching process, and the other is a knowledge of the PDP-11 instruction format.

The following facts pertain to the KD11-E/PDP-11 instruction set:

1. In general, the PDP-11 operation code is variable from 4 to 16 bits.

2. A number of instructions require two address calculations; an even larger number require only one address calculation. There are also a number of instructions that require address calculations, but do not operate on data.

3. All op codes that are not implemented in the KD11-E processor must be trapped.

4. There are illegal combinations of instructions and address modes that must be trapped.

5. There exists a list of exceptions in the execution of instructions having to do with both the treatment of data and the setting of condition codes in the processor status word.

4.5.2 Instruction Register

Each PDP-11 instruction obtained from memory is stored in the 16-bit instruction register (IR). This register consists of three 6-bit D-type 74174 registers (E55, E65, and E66 on K2-5) and one 74S74 D-type flip-flop (E33). The purpose of the IR is to store the instruction for the complete instruction cycle so that the IR Decode and Auxiliary ALU Control circuits can decode the correct control signals throughout the instruction cycle.

The IR latches data from the SSMUX 00–15 lines on K2-7 LOAD IR L and the leading edge of K1-5 PROC CLK L.

On the trailing edge of K2-9 BUT SERVICE (1) H, all the IR bits except K2-5 IR15 (1) H are cleared. [K2-5 IR15 (1) H is set by the same signal transition.] This means that the IR Decode circuit will see a conditional branch instruction in the IR after every *service* microstep. This action prevents the processor from decoding a HLT instruction after an Initialize condition.

If a bus error (BE) occurs while the Control Store output signal Enable Double Bus Error (K2-8 ENAB DBE L) is asserted, the whole IR is cleared (PDP-11 Halt), causing the processor to halt automatically. Bus errors occurring without the K2-8 ENAB DBE L signal have no effect on the IR. K2-8 ENAB DBE L is only asserted during certain microwords in the trap sequence to prevent the possibility of a second bus error occuring (Double Bus Error), which would cause the trap sequence to be re-entered before it is completed. For example, if R6 (Stack Pointer) were an odd address, the first bus reference using the stack in the trap routine would cause another trap (Odd Address), a sequence that could tie up the CPU indefinitely if not for the Halt and Double Bus Error facilities. In short, any bus error during the four memory references of the trap sequence is fatal.

4.5.3 Instruction Decoder

4.5.3.1 Instruction Decoder Circuitry - The Instruction Decode (prints K2-5 and K2-6) and Control Store (prints K2-7 through K2-10) circuitry could be thought of as an internal microprocessor that interprets PDP-11 instructions and translates them into a set of microinstructions, each consisting of 40 control signals. These control signals then determine the operation of the data path and Unibus control circuitry.

A block diagram of this internal microprocessor is shown in Figure 4-1. Note that all outputs of the Control Store ROMs (K2-7 through K2-10) are latched in hex D-type registers (74174s).

Nine of these latched signals (K2-7 MPC 08 H through K2-7 MPC 00 H) are fed back to the inputs of the Control Store ROM as the next microinstruction address (and can then be called the micro-PC). The wired-OR capability of these lines allows the IR Decode circuitry to force microbranching addresses on certain enabling conditions. The actual microbranch address will depend on the instruction being decoded, the instruction mode used (modes 0–7), and the operand required (source or destination).

The IR Decode circuitry is shown on prints K2-5 and K2-6. It consists of one 512 × 4 ROM, ten 256 × 4 ROMs, and two 32 × 8 ROMs, and 74H01, 7402, 7400, and 7410 logic gates. The following descriptions are based on instruction types. Complete block diagrams of the microcode flow are available in the KD11-E print set (drawing D-FD-KD11-E).

4.5.3.2 Double-Operand Instructions – Double-operand instructions require two address calculations, one for the source and one for the destination operand. The microbranch to the sequence of microinstructions that determine the source operand is initiated by the Control Store output signal K2-6 IR DECODE (1) H. When this signal is enabled, the IR Decode ROMs DOP Decode (E68 and E69 on print K2-6) check the instruction in the IR (op code bits IR15–12). If the instruction is a double-operand type, the ROM outputs are asserted as follows:

Type Instruction	K2-6 IR Code 00L	ROM Outputs K2-7 MPC07L	K2-7 MPC06L	K2-7 MPC05L	K2-7 MPC04L	K2-7 MPC03L
MOV (SM0*DM0)	1	1	0	0	0	1
DOP ($\overline{\text{MOV+SUB}}$) MOD (SM0*DM0) (ADD, BIC, BIS)	1	0	1	1	0	0
SUB ($\overline{\text{SM0*DM0}}$)	1	1	0	1	0	0
DOP (SM0*DM0)	1	0	1	0	0	1
Illegal Instructions	0	0	0	0	0	0
DOP NONMOD (SM0*DM0) (CMP, BIT)	1	1	0	1	1	1

NOTE
Ground on the MPC lines represents a logic "1."

Coupled with the microprocessor outputs of the DOP DEC ROM are the outputs of a set of type 74H01 gates on K2-6. These gates, when enabled, place the contents of the source mode field (IR11:09) of the PDP-11 instruction being decoded onto the MPC 00:02 lines. These gates are enabled by the K2-6 SRCH ROM output only when the instruction being decoded is of the double-operand type, the K2-6 IR DECODE (1) L signal is asserted, and the instruction is not reserved (K2-6 IR CODE 00 L unasserted).

A summary of the various source microaddresses is shown below:

Instruction	Source Mode	Octal Microbranch Address
DOP ($\overline{\text{SM0*DM0}}$)	0	110
	1	111
	2	112
	3	113
	4	114
	5	115
	6	116
	7	117
Reserved DOP		00

NOTE
A ground on the MPC lines represents a logic 1.

The DOP DEC ROMs described above are also used to decode the microprocessor address for the various Control Store destination operand routines. When the K2-7 BUT DEST L input is asserted by the miscellaneous control field circuitry of the Control Store, the DOP DEC ROMs decode the instructions, determine whether it is a modifying or nonmodifying instruction, and generate the following micro-PC addresses.

	ROM Outputs				
Type Instruction	**K2-7 MPC07L**	**K2-7 MPC06L**	**K2-7 MPC05L**	**K2-7 MPC04L**	**K2-7 MPC03L**
Move ($\overline{SM0*DM0}$)	0	0	1	0	1
Modify (ADD BIS BIC but not MOV or SUB)	0	0	1	1	1
Nonmodify (CMP BIT)	0	0	1	1	0
SUB	0	0	1	0	0

The circuitry used to decode the destination mode field of the instruction being decoded is similar to that described above for microaddressing the source operand routine. A set of 74H01 gates on K2-6 is used to place the contents of K2-5 IR 05 (1) H through K2-5 IR 03 (1) H on the lines when enabled. For double-operand instructions, enabling occurs when the MPC miscellaneous control field asserts K2-7 BUT DEST L.

ROM E73 on print K2-6 is also considered to be part of the DOP Decoder circuitry. This ROM decodes all Extended Instruction Set (EIS) instructions, generating the following micro-PC addresses when K2-6 IR DECODE (1) H is asserted:

		ROM Outputs				
Type Instruction	**K2-6 IR Code 00L**	**K2-7 MPC07L**	**K2-7 MPC06L**	**K2-7 MPC05L**	**K2-7 MPC04L**	**K2-7 MPC03L**
Multiply or Divide (MUL, DIV)	1	1	0	0	1	0
Arithmetic Shift or Arithmetic Shift Combined (ASH,ASHC)	1	1	0	0	1	1
SOP	1	1	0	1	1	0
XOR	1	0	1	0	0	1
Reserved	0	0	0	0	0	0

The K2-6 DEST L output of the EIS Decoder ROM (E73) allows the 74H01 (E64) on print K2-6 to place the contents of the destination mode field of the instruction being decoded onto the micro-PC (MPC00-MPC02) lines. This microbranching technique is similar to that described above for micro-addressing the source operand routine. Use of the EIS instructions does not degrade processor timing or affect NPR latency.

4.5.3.3 Single-Operand Instructions - Unlike double-operand instructions, single-operand instructions only require one address calculation to obtain the necessary operand. Complete SOP instruction decoding is done with the two 256- × 4-bit ROMs (E59 and E58).

The SOP Microbranch ROM (E59) monitors the necessary IR input lines and asserts the correct micro-PC address on lines K2-7 MPC03-L through K2-7 MPC 06 L when the K2-6 IR DECODE (1) L signal is asserted and the SOP enable signal K2-5 IR 12-14=0 H is true. The K2-6 DEST L output is also activated when an SOP instruction is decoded. This signal enables the destination mode monitoring circuitry described in the double-operand instruction decoding section. Microaddresses for SOP instructions are shown below.

Instruction	Base Microbranch Address
SOP Modify (CLR,COM,INC,DEC)	040
SOP Non-Modify (TST)	160
NEG	150
Rotate and Shift	170
JSR	150
JMP	020
MARK	
SWAB	030
MFPI (D)	100
MTPI (D)	250
MFPS	130
MTPS	120

The SOP Microbranch ROM (E59) is also used to decode JSR instructions. This decoding is performed in the same manner as that for SOP instructions. The K2-6 DM0 H input to the ROM is used to detect the illegal instruction JMP or JSR destination mode 0. When this occurs, no micro-PC address is allowed on the ROM outputs.

The SOP Decode ROM (E58) monitors the same input signals as the SOP Microbranch ROM. Its purpose, however, is to decode illegal, reserved, and trap instructions. The three output signals K2-6 IR CODE 00 L through K2-6 IR CODE 02 L are enabled as follows:

Instructions	IR Code 02	01	00
Reserved	1	1	0
Illegal (JMP or JSR Mode 0)	1	0	1
EMT	0	1	0
Trap	0	0	1

The fourth output signal of the SOP Decdoe ROM enables the destination mode monitoring circuitry described in the double-operand instruction decoding section.

4.5.3.4 Branch Instructions – Conditional branch instructions are completely decoded by the Branch DEC ROM (E71 on print K2-6). This ROM is enabled when bits IR11:IR14 are all low and the K2-6 IR DECODE (1) L signal is active. The input lines monitored are the four condition code bits (N, Z, V, and C) and four IR bits (IR15, 10, 9, and 8). When a branch is decoded, the K2-7 MPC 07 L output signal is enabled. The branch instruction microcode routine in the Control Store will sign-extend the branch offset and shift it left one place.

4.5.3.5 Operate Instructions – There are three 256- × 4-bit ROMs in the instruction-decoding circuitry for decoding PDP-11 operate instructions. These ROMs are the Reset/Trap Decode, Trap Decode, and Op Branch ROMs (E62), all found on K2-6.

The Op Branch ROM (E62) monitors IR output lines IR00:IR07. It is enabled when IR08 and IR15 are low and K2-6 IR DECODE (1) L is active. The PDP-11 operate instructions are decoded into the following micro-PC addresses on the ROM outputs K2-7 MPC 00 L through K2-7 MPC 03 L.

Instruction	Microbranch Address
Reset	003
RTI/RTT	011
Set Condition Codes	007
Clear Condition Codes	006
RTS	004
Wait	014

The Reset/Trap Decode ROM (E53) decodes Reset, RTT, and RTI instructions and activates the outputs K2-6 START RESET H and K2-6 ENAB TBIT H accordingly. This ROM also allows the lower PSW bits (K2-6 DISABLE LOAD PSW H) to be loaded only from the stack when the processor is operating in User mode (memory management restriction). It also treats a Reset instruction as a NOP in User mode.

The TRAP DEC ROM (E52) has the same inputs as the Op Branch ROM. Its purpose is to decode Halt, reserved, trap, and illegal instructions, and to enable the outputs accordingly. The K2-3 USER MODE H input also allows this ROM to treat Halt instructions as reserved instructions when operating in the memory management User mode.

	IR Code		
Instruction	**02**	**01**	**00**
Reserved	1	1	0
Illegal	1	0	1
BPT	1	0	0
IOT	0	1	1
HALT	Enable HLT RQST L		

4.6 AUXILIARY ALU CONTROL

The AUX Control circuitry on the KD11-E consists of three bipolar ROMs, shown on K2-5.

ROM	**Name**
32- × 8-bit	DOP (E81)
256- × 4-bit	SOP (E60)
256- × 4-bit	ROT/SHIFT (E61)

These ROMs determine the ALU operation to be performed whenever the microcode executes the action X ← Y OP B, where Y designates a scratchpad register and X designates either the B REG or a scratchpad register.

The AUX DOP ROM (E81) decodes double-operand instructions, and is enabled by K2-8 AUX SETUP H. The following table expresses the outputs of this ROM as a function of the instruction being performed. (B represents the B register, A represents any scratchpad register, and F represents the ALU output.)

		ROM Outputs			
Instruction	**ALU Operation**	**Func Code 03H**	**Func Code 02H**	**Func Code 01H**	**Func Code 00H**
MOV (B)	F ← A	0	1	0	1
COMP (B)	F ← A minus B	0	1	0	0
ADD	F ← A plus B	1	0	0	0
SUB	F ← A minus B	0	1	0	0
BIT (B)	F ← $\underline{A}$·B	1	0	0	1
BIC (B)	F ← A·B	1	0	1	0
BIS (B)	F ← A+B	1	0	1	1
XOR	F ← A ⊕ B	1	1	0	0

The AUX SOP ROM (E60) decodes single-operand instructions, and is enabled by K2-8 AUX SETUP H. The following table expresses the ROM outputs as a function of the SOP instruction decoded.

Instruction	ALU Function	ROM Outputs Func Code 03H	Func Code 02H	Func Code 01H	Func Code 00H
SWAB	F ← A	0	1	0	1
CLR (B)	F ← ZERO	0	0	0	0
COM (B)	F ← $\overline{A}$	0	0	0	1
INC (B)	F ← A plus 1	0	0	1	0
DEC (B)	F ← A minus 1	0	0	1	1
NEG (B)	F ← A minus B	0	1	0	0
ADC (B)	F ← A plus CBIT (0)	0	1	0	1
	F ← A plus CBIT (1)	0	0	1	0
SBC (B)	F ← A minus CBIT (0)	0	1	0	1
	F ← A minus CBIT (1)	0	0	1	1
TST (B)	F ← A	0	1	0	1
ROR (B)	F ← B	0	1	1	0
ROL (B)	F ← B	0	1	1	0
ASR (B)	F ← B	0	1	1	0
ASL (B)	F ← B	0	1	1	0
MARK	N/A	0	0	0	0
MFPI	F ← A	0	1	0	1
MTPI	F ← A	0	1	0	1
SXT	F ← NBIT (0)	0	0	0	0
	F ← NBIT (1)	0	1	1	0
MTPS	F ← A	0	1	0	1
MFPD	F ← A	0	1	0	1
MTPD	F ← A	0	1	0	1
MFPS	F ← A	0	1	0	1

Auxiliary control signals are also necessary for performing rotate and shift operations. The ROT/SHFT ROM (E61) on K2-5 decodes these instructions and outputs those control signals required to shift the contents of the B REG. Inputs K1-1 BREG 00 (1) H, K1-10 CC N H, and K1-1 CBIT (1) H also determine the K2-5 SERIAL SHIFT H and K2-5 ROT CBIT (1) H signals. The SERIAL SHIFT H signal is sent to the BYTE MUX (E106 on K1-10), where it is used in determining the K1-10 SHIFT IN 07 H signal used in the B REG shifting operation. K2-5 ROT CBIT (1) H is used in the calculation of the new carry condition (C and V Bit ROM – E105 on K1-10). Note that for all rotate and shift operations, the AUX SETUP is performed on the B ← B step before each X ← Y OP B step previously mentioned. This is done to allow the condition codes to be set up without slowing the processor.

Table 4-9 summarizes the auxiliary control instructions.

Table 4-9 Auxiliary Control for Binary and Unary Instructions

Instruction	N and Z	Condition Codes		ALU Function	CIN
		V	C		
MOV(B)	Load	Cleared	Not affected	A Logical	0
CMP(B)	Load	Load like Subtract.	Load like Subtract.	A minus B	0
BIT(B)	Load	Cleared	Not affected	A • B Logical	0
BIC(B)	Load	Cleared	Not affected	$\overline{A}$ • B Logical	0
BIS(B)	Load	Cleared	Not affected	A ← B Logical	0
ADD	Load	Set if operands are same sign and result different.	Set if carry out.	A plus B	0
SUB	Load	Set if there was arithmetic overflow as a result of the operation (i.e., if operands were of opposite signs and the sign of the source was the same as the sign of the result; cleared otherwise.	Set if carry.	A minus B	0
XOR	Load	Cleared	Not affected	A * B	0
CLR(B)	Load	Cleared (like Add)	Clear	0	0
COM(B)	Load	Cleared	Set	$\overline{A}$	0
INC(B)	Load	Set if destination held 100000 before operand.	Not affected	A plus 1	+1
DEC	Load	Set if result is 100000.	Not affected	A minus 1	1
NEG(B)	Load	Set if result is 100000.	Cleared if result is 0; set otherwise.	A minus B	0
ADC(B)	Load	Set if destination was 077777 and C = 1.	Set if destination was 177777 and C = 1.	A plus CBIT	0
SBC(B)	Load	Set if destination was 100000.	Set if destination was 0 and C = 1; cleared otherwise.	A minus CBIT	0
TST(B)	Load	Cleared	Cleared	A Logical	0
ROR(B)	Z←1 If(15:01)*C = 0 N←C	Unaffected	(0)	B Logical	0

Table 4-9 Auxiliary Control for Binary and Unary Instructions (Cont)

Instruction	N and Z	Condition Codes		ALU Function	CIN
		V	C		
ROL(B)	Z←1 If(14:00)*C = 0 N←(14)	Unaffected	(15) B (7)	B Logical	0
ASR(B)	Z←1 If(15:01) = 0 N←N	Unaffected	O ← (15)	B Logical	0
ASL(B)	Z←1 If(14:01) = 0 N←(14)		C ← (15)	B Logical	0
SWAB	Load	Cleared	Cleared	A Logical	0
SXT	Z–Load N–Unaffected	Cleared	Cleared	1	0
MFPI	Load	Cleared	Unaffected	A Logical	0
MTPI	Load	Cleared	Unaffected	A Logical	0
MTPS	Z–Set If SRC(7)=0 N–Set If SRC(7)=1	Cleared	Unaffected	A Logical	0
MFPD	Load	Cleared	Unaffected	A Logical	0
MTPD	Load	Cleared	Unaffected	A Logical	0
MFPS	Z–Set If PS(7)=0 N–Set If PS(7)=1	Cleared	Unaffected	A Logical	0

4.7 DATA TRANSFER CIRCUITRY

4.7.1 General Description

All Unibus data transfers are controlled by the DAT TRAN circuitry on K2-1. This logic monitors the busy status of the Unibus, controls the processor bus control lines BBSY, MSYN, C1, and C0, and detects parity errors (PE), and bus errors (BE).

4.7.2 Control Circuitry

4.7.2.1 Processor Clock Inhibit – All processor data transfers on the Unibus are initiated by K2-8 BUF DAT TRAN (1) H. When K1-5 TAP 30 H goes high, the signal combines with the signal K2-1 EOT L (normally a logic 1 between transfers) to create K2-1 TRAN INH L, shutting off the processor clock until the transfer is completed.

4.7.2.2 Unibus Synchronization – The synchronizer logic shown in Figure 4-19 (from K2-1) arbitrates whether the processor or some other Unibus peripheral will control the Unibus. A logic 1 level (+3 V) at the set input of the E31 flip-flop on K2-1 specifies that the bus is presently in use. Each of the inputs that combine to create this level monitors a specific set of bus conditions.

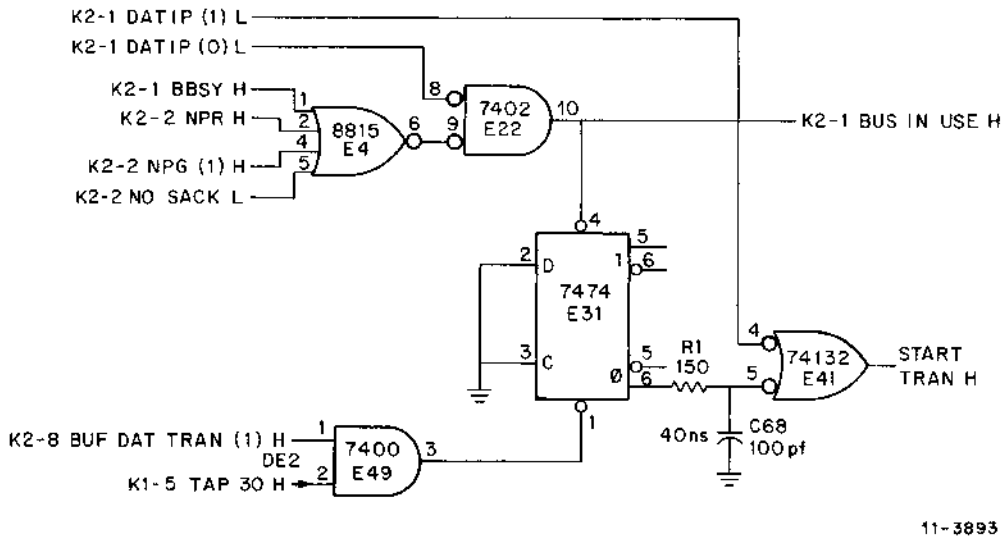


Figure 4-19 Unibus Synchronizer

NPR (K2-2 NPR H)	A Unibus peripheral has asserted a nonprocessor request (NPR) and wishes to gain control of the bus immediately.
BBSY (K2-1 BBSY H)	Another Unibus peripheral already has control of the bus, and is asserting a bus busy (BBSY) signal.
NPG [K2-2 NPG (1) H]	An NPR device has requested control of the Unibus and the KD11-E processor has issued a nonprocessor request grant (NPG). The condition may exist where the NPR device has already recognized the NPG and has dropped its NPR signal, while not yet having asserted a SACK or BBSY.
NO SACK L (K2-2 NO SACK)	A device has requested control of the Unibus. The KD11-E processor has issued a grant, and the device has returned SACK L, causing NO SACK L to go high. The condition may exist where only SACK L remains on the Unibus for a period of time before the peripheral asserts BBSY.
DATIP (0) L [K2-1 DATIP (0) 1]	When this input is true, all of the above signals are overridden. It indicates that the processor is performing a DATIP (Read/Modify/Write) operation, and has control of the Unibus (BBSY asserted). NPR devices may, however, be granted bus control, but must wait until the processor releases BBSY before asserting theirs. (DATIP operations dictate worst-case bus latencies for NPR devices.)
BUS SSYN L	A data transfer is still being completed; therefore, the processor must wait before initiating another.

If none of the above Bus-in-Use conditions exist, the K2-8 BUF DAT TRAN (1) H signal sets the E31 flip-flop on K2-1 when K1-5 TAP 30 H goes high, and activates K2-1 START TRAN H to start the transfer. The RC circuit at the output of E36 eliminates any noise that may result from the synchronizer under worst-case conditions.

4.7.2.3 Bus Control - Once the K2-1 START TRAN H signal is activated, the DAT TRAN circuitry begins a Unibus data transfer operation by asserting K2-1 ENAB ADDRS L, triggering the following actions:

1. Enables the bus address drivers (BUS A15:A00 on K1-6).

2. Enables the BBSY driver (K2-1).

3. Enables the bus control signals BUS C0 and BUS C1, which determine the kind of transfer being performed.

C1	C0	Operation
0	0	DATI
0	1	DATIP
1	0	DATO
1	1	DATOB

The actual condition of these control lines is determined by K2-8 BUF C0 (1) H and K2-8 BUF C1 (1) H.

4. Enables the bus data drivers (BUS D00–BUS D15) if the operation being performed is a DATO.

4.7.2.4 M8264 NO-SACK Timeout Module – The M8264 is a quad-height module containing circuitry that asserts BUS SACK L on the Unibus if a device requesting Unibus control does not assert SACK within 10 μs after a grant line has been enabled (Figure 4-20).

The grant signals (BUS NPG and BUS BG7 through BUS BG4) are ORed (E3) on the M8264. The output of the OR gate enables a NAND gate (E6) and triggers a monostable multivibrator (E5). The signals produced are ANDed (E1) to enable BUS SACK L. The monostable effectively delays (by 10 μs) the assertion of BUS SACK L since it produces a 10 μs pulse which prevents the AND gate from being enabled. BUS SACK L, when asserted, will cause the processor to drop the grant line, which will in turn cause the M8264 to drop BUS SACK L.

This module prevents the processor from being hung if a grant line is asserted and BUS SACK is not returned by the device requesting bus control. If the requesting device returns BUS SACK, the M8264 will not assert BUS SACK since the grant line will be dropped before the monostable times out.

As a maintenance aid, a counter is provided (E4) to drive a set of LEDs. The binary counter counts up one each time BUS SACK is asserted by the M8264 and the number of occurrences is indicated by the LEDs. The maximum number recorded is 15 before the counter is reset. The counter is cleared and the monostable multivibrator is reset when BUS DC LO L becomes asserted (i.e., upon system power-up).

4.7.2.5 MSYN/SSYN Time-Out Circuitry – Unibus specifications require that the BUS MSYN L control signal be enabled no sooner than 150 ns after the bus address, data, and control lines have been asserted. To meet this requirement, the circuitry in Figure 4-21 has been incorporated into the DAT TRAN logic (K2-1).

The multiplexer (E10) shown in Figure 4-22 helps adapt the DAT TRAN circuitry to the type of bus operation being performed (DATI or DATO). Specific functions performed are as follows:

1. Generates the correct Unibus control signals [K2-1 UBUS CO (1) H and K2-1 UBUS C1 (1) H].

2. Inhibits the detection of parity errors during DATO operations.

3. Generates an End of Transfer (EOT L) signal as soon as BUS SSYN is returned by an addressed peripheral.

4. Delays the assertion of BUS MSYN, using the clock signal K1-5 ALLOW MSYN H, which does not become asserted until the Physical Bus Address register has been loaded.

NOTE

This applies only to DATI or DATIP. During DATO or DATOB, the bus address is never loaded in the same microcycle that does the DATO or DATOB.

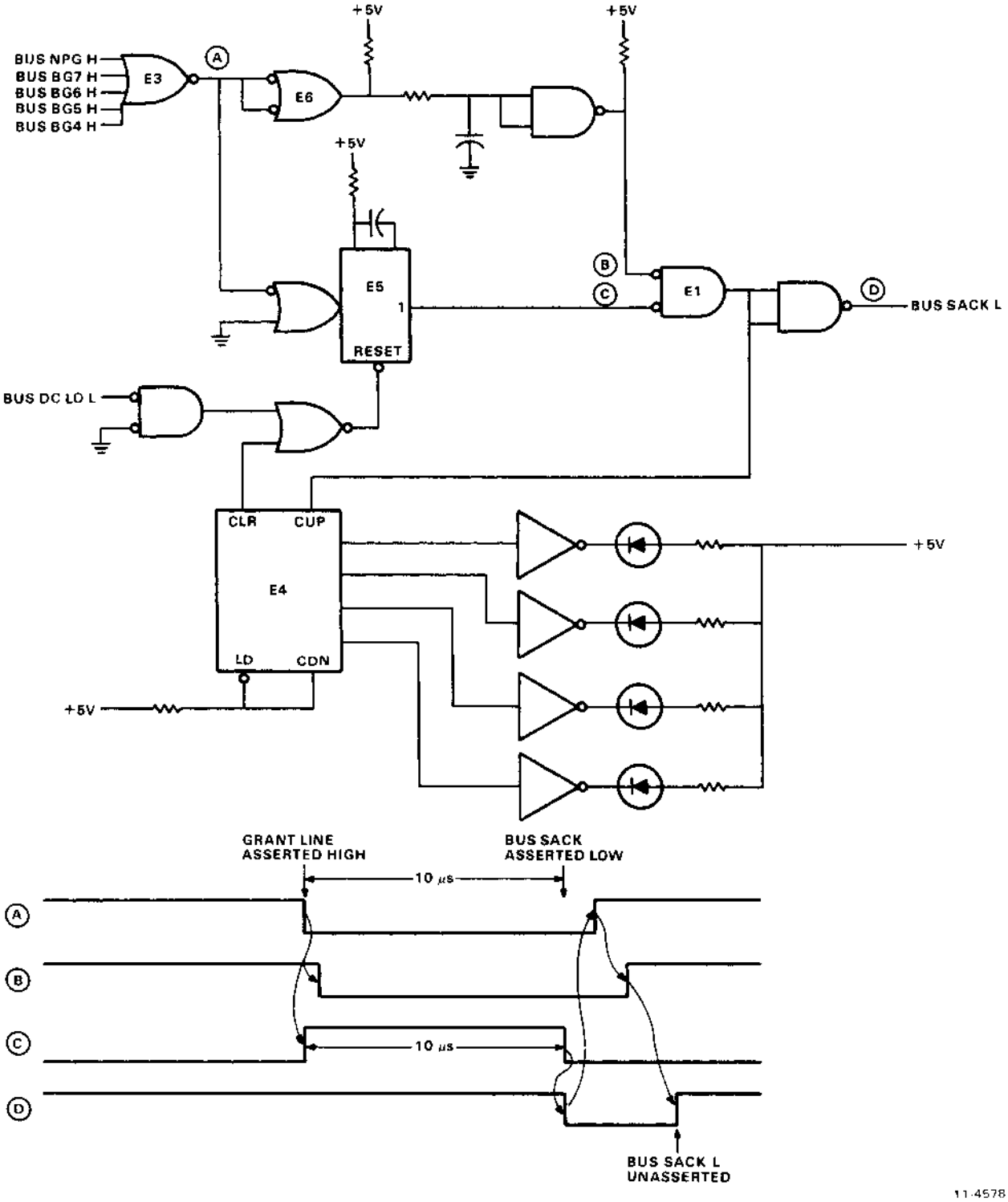


Figure 4-20 NO-SACK Timeout Module

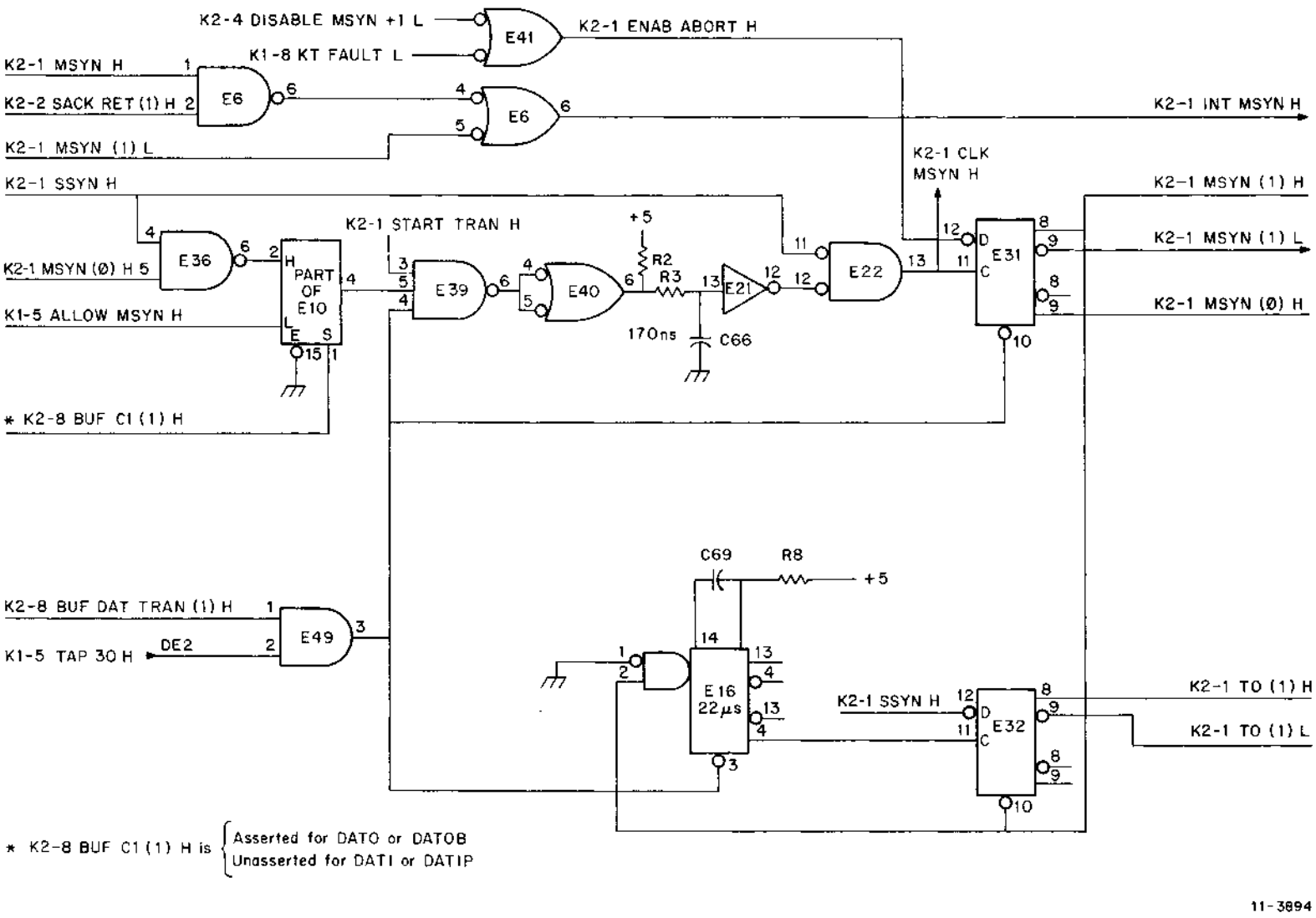


Figure 4-21 SSYN/MSYN Control

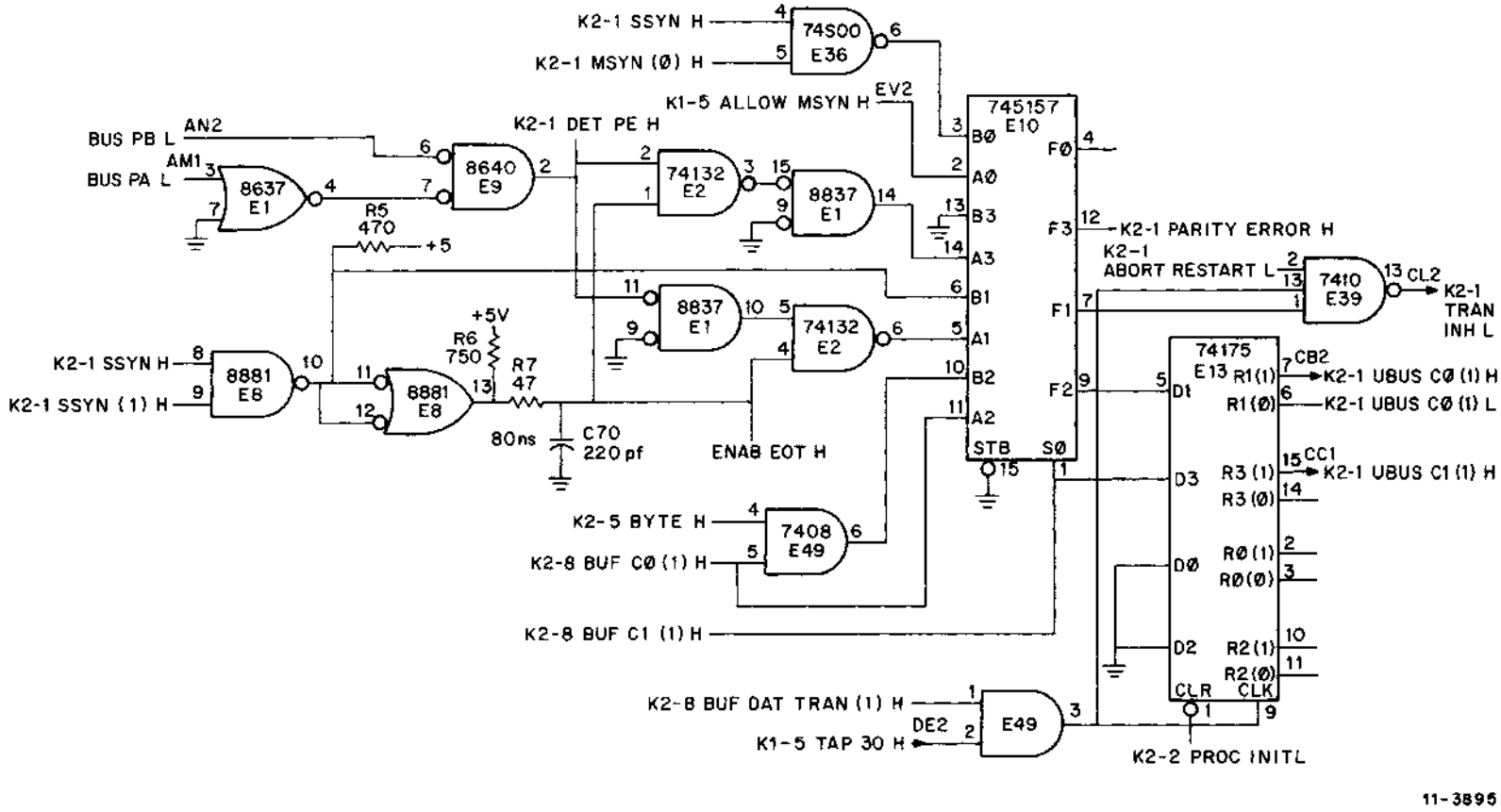


Figure 4-22 Data Transfer Multiplexer

The RC circuit shown in Figure 4-21 prevents the MSYN flip-flop (E31) from being clocked until approximately 150 ns after the bus address and control lines are placed on the bus. Once this latch is set, BUS MSYN L is activated and the SSYN TIMEOUT one-shot E16 is triggered. When SSYN is returned by the addressed peripheral, both the MSYN flip-flop (E31) and the SSYN TIMEOUT one-shot are cleared. The processor clock is then freed by the release of K2-1 TRAN INH L. If a DATI or DATIP operation is being performed, that will be clocked into either the scratchpad, B REG, PSW, or IR on the next low-to-high transition of K1-5 PROC CLK L.

If a DATO or DATOB operation is being performed, the data bus drivers are disabled after SSYN is returned from the addressed peripheral but before the MSYN line is unasserted.

4.7.2.6 Bus Errors – Once the SSYN TIMEOUT one-shot is triggered, SSYN must be returned within 22 μs. If SSYN is not returned in this time, E16 times out, setting the TIMEOUT flip-flop (E32). The output of this latch then generates the signal K2-1 ABORT RESTART L and pulse K2-1 ABORT H. K2-1 ABORT RESTART L reenables the PROC CLK and K2-1 ABORT H sets the Bus Error flip-flop (E33). This same pulse that sets the Bus Error flip-flop also clears the micro-PC address latches (MPC00 through MPC008) on K2-7, forcing the processor to enter the service microroutine on the next PROC CLK L low-to-high transition.

4.7.2.7 Parity Errors – If a data transfer is being performed with a parity memory (e.g., MS11-JP or MM11-DP), all parity errors detected by the memory will be reflected back to the KD11-E on the Unibus lines BUS PA L and BUS PB L on K2-1 (Figure 4-23).

Control		Error
PA	**PB**	**Description**
0	0	No Parity Error
0	1	Parity Error on DATI
1	0	Reserved for future use
1	1	Reserved for future use

Errors detected while performing a DATIP or DATI [K2-8 BUF C1 (1) H unasserted] will result in the Parity Error flip-flop (E34) being set when SSYN is returned to the processor. Processor operations resulting from Parity Error will be discussed further in Paragraph 4.11, Service Traps.

4.7.2.8 End of Transfer Circuitry – To synchronize the DAT TRAN logic with the main KD11-E processor clock, the End of Transfer (EOT) circuitry (Figure 4-24) has been incorporated into the CPU (K2-1). During a DATI or DATIP, an EOT L signal is generated approximately 100 ns after SSYN is returned to the processor. That EOT L removes the processor clock disabling signal (Paragraph 4.7.2.1), K2-1 TRAN INH L. During a DATO or DATOB, K2-1 TRAN INH L is unasserted immediately when SSYN is returned.

4.7.2.9 Data-in-Pause Transfer – Another circuit included in the DAT TRAN logic detects Data-in-Pause (DATIP) transfers and controls the bus control signal BBSY. When a DATIP (Read/Modify/Write) bus operation is initiated, the flip-flop (E32) is latched, forcing the processor to hold BBSY L until the DATO portion of the routine has been completed. While BBSY is asserted, no other Unibus peripheral can seize control of the bus. This feature often determines the maximum bus latency for NPR devices (K2-1).

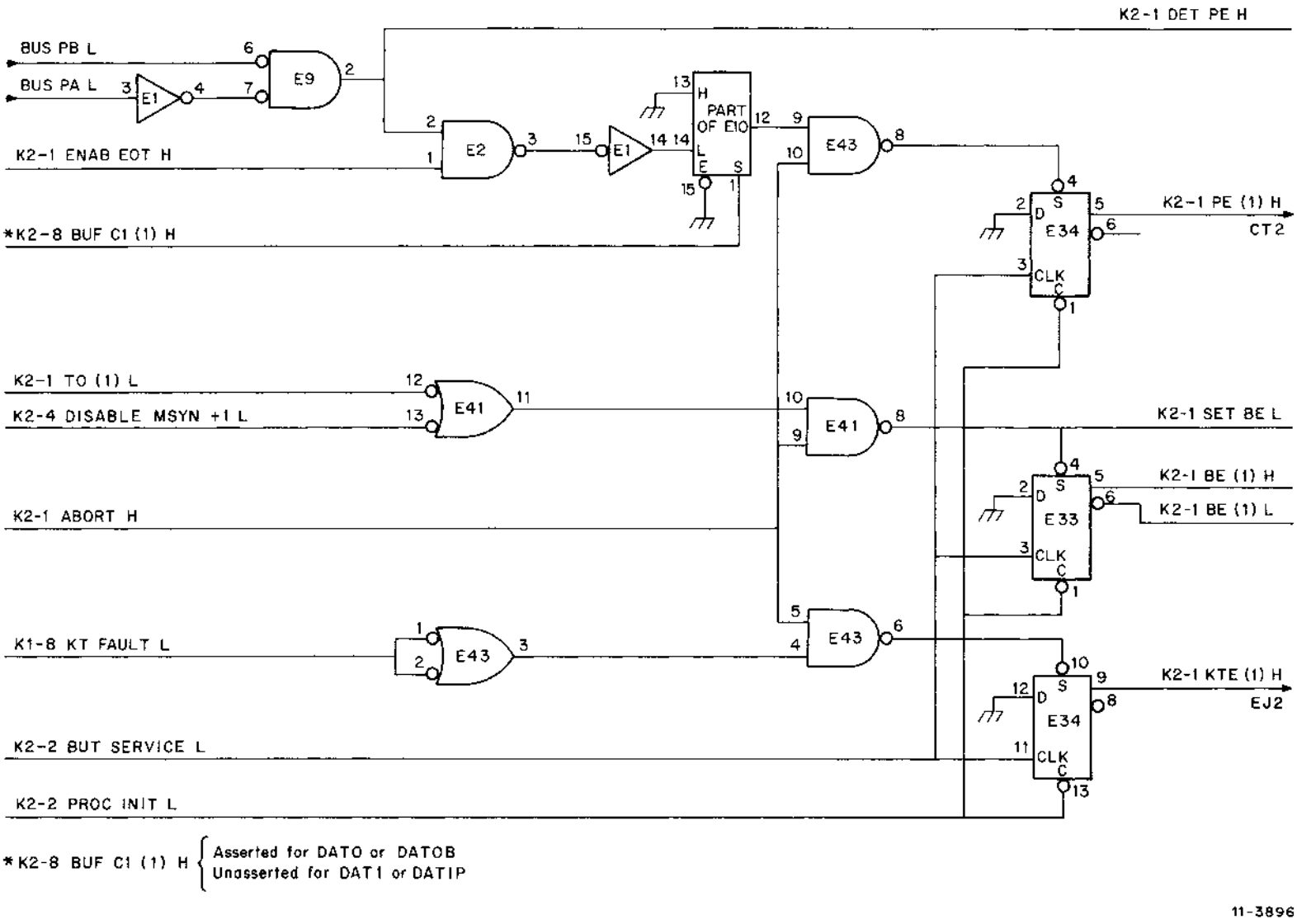


Figure 4-23 Error Logic

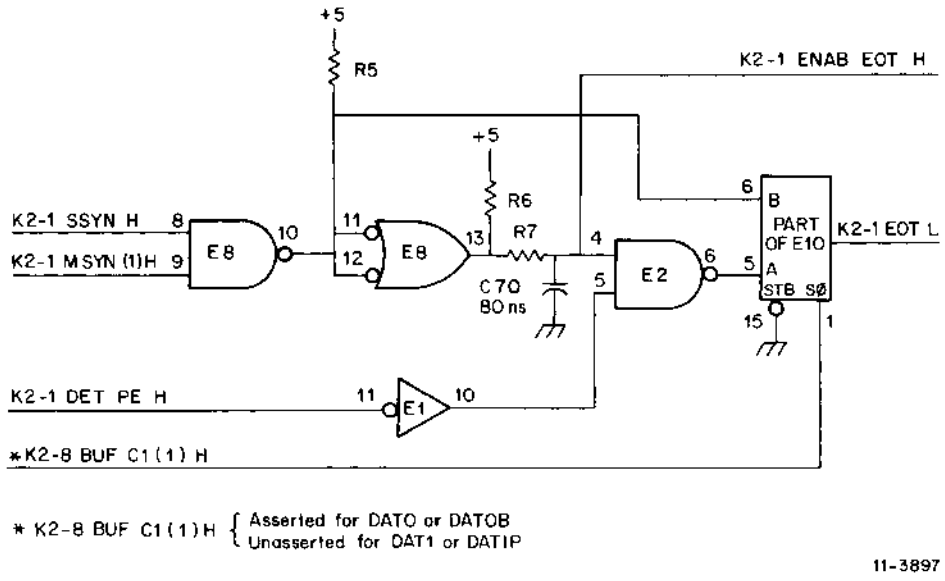


Figure 4-24 End-of-Transfer Logic

4.7.2.10 Odd Address Detection - The circuitry shown in Figure 4-25 is incorporated in the KD11-E to detect odd address errors. ROM E78 (print K2-8) monitors the signals K2-8 BUF DAT TRAN (1) H, K2-5 BYTE H, and K1-6 VBA00 (1) H, and asserts K2-8 DISABLE MSYN L when an odd address is detected. The multiplexer circuit (E38 on K2-4) forces the processor to always autoincrement or autodecrement the PC (R7) or the SP (R6) scratchpad registers by two, regardless of the type of instruction being performed. This is done by preventing the K2-4 DISABLE MSYN +1 L signal from being asserted.

4.8 POWER FAIL/AUTO RESTART

The KD11-E power fail/auto restart circuitry (K2-3) serves the following purposes:

1. Initializes the microprogram, the Unibus control, and the Unibus to a known state immediately after power is applied to the computer.

2. Notifies the microprogram of an impending power failure.

3. Prevents the processor from responding to an impending power failure for 2 ms after initial startup.

The actual power fail/auto restart sequences are microprogram routines. The operation of the power fail/auto restart circuitry depends on the proper sequencing of two bus signals: AC LO and DC LO. Because of the electrical properties of the Unibus drivers and receivers, the entire computer system must be powered up for the machine to operate. Therefore, the processor is notified of a power fail in peripherals, as well as in its own ac source.

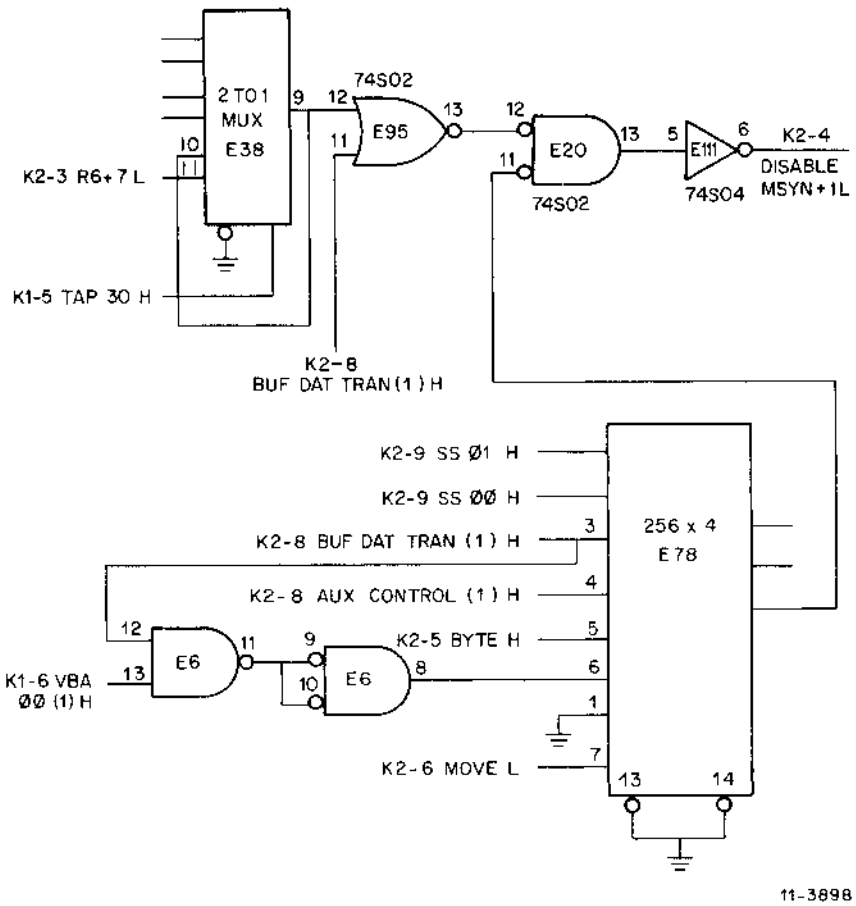


Figure 4-25 Odd Address Detection

The notification of power status of any PDP-11 system component is transmitted from each device by the signals BUS AC LO L and BUS DC LO L (K2-3). The power-up sequence (Figure 4-26) shows that BUS DC LO L is unasserted before BUS AC LO L is unasserted. When BUS DC LO L is not asserted, it is assumed that the power in every component of the system is sufficient to operate. When BUS AC LO L is not asserted, there is sufficient stored energy in the regulator capacitors of the power supply to operate the computer for 5 ms, should power be shut down immediately.

As ac power is removed, BUS AC LO L is asserted first by the power supply warning the processor of an impending power failure. When BUS DC LO L is asserted, it must be assumed that the computer system can no longer operate predictably. Memories manufactured by DIGITAL use BUS DC LO L as a switched signal, turning them off even if power is still available. Time $\Delta t2$ (Figure 4-26) is the time delay between the assertion of BUS AC LO L and the assertion of BUS DC LO L; this time delay must be greater than 5 ms. This allows for power to be rapidly cycled on and off. According to PDP-11 specifications, upon system startup a minimum of 2 ms run time is guaranteed before a power fail trap occurs, even if the line power is removed simultaneously with the beginning of the power-up sequence. After the power fail trap occurs, a minimum of 2 ms run time is guaranteed before the system shuts down. Given the tolerances permitted in the timing circuitry used in most in most equipment, $\Delta t2$ must be greater than 5 ms.

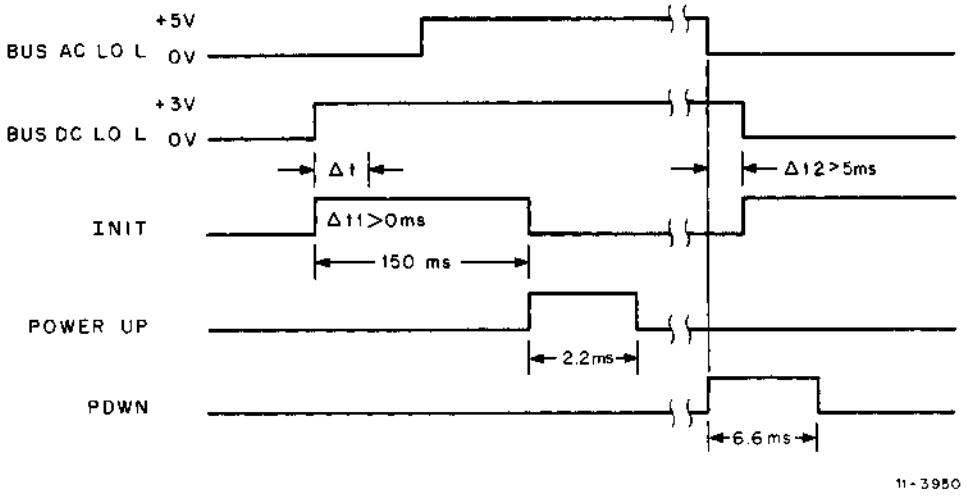


Figure 4-26 BUS AC LO and BUS DC LO Timing Diagram

When a pending power fail is sensed, a program trap occurs, causing the present contents of PC (R7) and the PSW to be pushed onto the memory stack, as determined by the contents of R6 (Stack Pointer register). The PSW is then loaded with the contents of location 26_8 and R7 with the contents of 24_8. Processing is continued with the new R7 and PSW. The user's program must prepare for the impending power failure by storing away volatile registers and reloading location 24_8 and 26_8 with a power-up vector. This vector points to the beginning of a restart routine.

When power is restored, the processor loads the PC (R7) with the contents of location 24_8 and the PSW with the contents of location 26_8. After loading these registers, the user program presumably will prepare locations 24_8 and 26_8 for another power failure. If the HLT RQST L input is asserted by an external switch closure, the processor powers up through locations 24_8 and 26_8, and halts.

Schematics for the power fail, auto restart, and bus reset logic are on K2-3. One-shot E14 generates a 150-ms processor INIT pulse as soon as BUS DC LO L is nonasserted after power is applied to the processor. At the end of 150 ms, the PUP one-shot (E7) is fired if BUS DC LO L is not asserted and the processor begins the PC and PSW load routine. The PUP one-shot generates a 2-ms pulse, during which the assertion of BUS AC LO L is ignored.

The triggering of the 150-ms INIT one-shot also resets the POWER INIT flip-flop (E24). Setting this flip-flop forces the Control Store to run the power-up routine beginning at micro-PC address 001. It is this routine that reads locations 24_8 and 26_8 for the new PC and PSW.

After PUP has timed out, the assertion of BUS AC LO L would fire the one-shot PDWN (E7). Upon entering the next service microcode state, K2-3 PFAIL H is latched into E19 (K2-2), causing a power fail trap to be recognized by the microprogram on entering the next service state. Various traps are arbitrated by the BUT service ROMs (E50 and E51 on K2-3).

If a momentary power failure occurs that causes the assertion of BUS AC LO L but does not cause the assertion of BUS DC LO L, the processor will restart when the PDWN one-shot times out, retriggering the INIT one-shot.

When a Reset instruction is decoded by ROM E53, the ROM output signal START RESET H is clocked into the Start Reset flip-flop (E54 on K2-2). This flip-flop output triggers a 100-ms INIT, after which the processor continues operation.

4.9 PROCESSOR CLOCK

The processor clock circuitry for the KD11-E is shown in Figure 4-27 and on print K1-5. A delay line is used to generate a pulse train, to which the entire processor is synchronized. Because the KD11-E is a fully clocked processor, events that result in the alteration of storage registers occur only on defined edges of the processor clock.

If all clock disable inputs are unasserted, the clock will begin running as soon as +5 V is applied. The length of an operating cycle can be either 180 ns or 240 ns, depending on the nature of the instruction being performed. Most microinstructions employ the shorter cycle, with the longer one only necessary when the machine is performing a DATO or DATOB, or in situations where the condition code must be determined before an operation can be performed. Long cycles are also used in loading the Bus Address register when memory management is turned on.

The clock is turned on and off by the gating of the feedback through the delay line. Taps of 120 ns, 90 ns, and 30 ns make it possible to vary the length of the cycle, according to a signal input [K2-8 LONG CYCLE (1) L] from the Control Store, as the processor clock timing diagrams in Figure 4-17 show. The indicated jumpers are inserted at W1 and W2 in the standard configuration; the overall cycle can be slowed down slightly (approximately 30 ns) by inserting jumpers at the alternate locations (shown on K1-5 by dotted lines) instead. It is also possible to disable the clock manually and use the manual clock input; any TTL-compatible waveform may be employed. Multiplexer E94 issues the feedback signal that, in effect, determines the length of the cycle.

The clock is turned off by the appropriate signal under the following conditions:

1. During a BUS INIT that is not caused by a RESET
2. During the INIT portion of the power-up routine
3. During the INIT portion of the power-down routine
4. During a Reset
5. During the BUT Service arbitration delay
6. During a priority interrupt
7. While BUS SACK is asserted by an interrupting device (not for NPR transfers)
8. During bus data transfers
9. After a Halt instruction is executed
10. When the manual clock is enabled

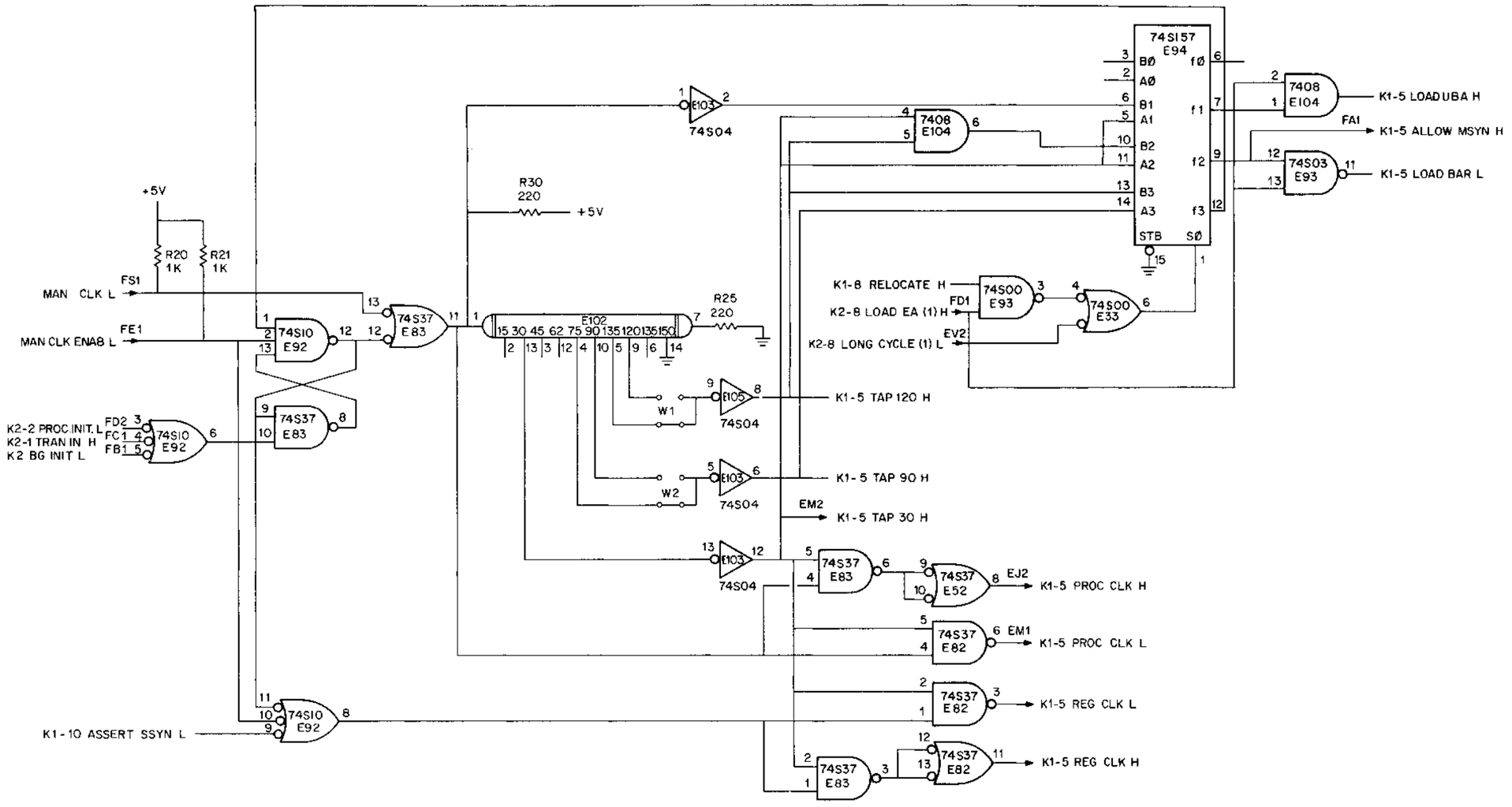


Figure 4-27 Processor Clock Circuit

4.10 PRIORITY ARBITRATION

4.10.1 Bus Requests

The KD11-E responds to bus requests (BRs) in a manner similar to that of the other PDP-11 processors. Peripherals may request the use of the Unibus in order to make data transfers or to interrupt the current processor program by asserting a signal on one of the four BR lines, numbered BR4, BR5, BR6, and BR7 in order of increasing priority. For example, if two devices, one at priority 5 and the other at priority 7, assert BRs simultaneously, the device at priority 7 is serviced first. Furthermore, if the processor priority, determined by PSW bits 07:05, is at level 4, only devices requesting BRs at levels higher than 4, such as BR7, BR6, or BR5, are serviced. Table 4-10 contains the order of priority for all BRs and other traps.

Table 4-10 Priority Service Order

Priority	Service Order
Highest	Halt Instructions
	Odd Address
	Memory Management Error
	Time-Out
	Parity Error
	Trap Instruction
	Trace Trap
	Stack Overflow
	Power Fail
	Halt from Console
	BR7
	BR6
	BR5
	BR4
Lowest	Next Instruction Fetch

Because a BR can cause a program interrupt, it may be serviced only after completion of the current instruction in the IR. A device that requests a program interrupt must, at the appropriate time, place a vector address on the Unibus data lines. The processor first stacks away the current contents of PSW and R7; then a new PSW is loaded from the contents of the vector address plus two and a new PC is loaded with the contents of the vector address. Further discussion of how the processor handles this BR routine is contained in the section on Service (Paragraph 4.11).

Arbitration logic for BRs is contained on print K2-2 and in Figure 4-28. All BRs are received directly from the Unibus (Unibus receivers E17), and latched into register E19 (quad D-type latch, 74S174) when the microprogram enters the next service state [K2-9 BUT SERVICE (1) H is true]. The BR Priority Arbitration ROM (E29) then determines whether the present processor priority [PSW (7:4)] is higher than the highest BR received and, if not, which BR received has the highest priority. Arbitration performed by E29 in the order of priority are shown below.

HLT RQST
PSW7
BR7
PSW6
BR6
PSW5
BR5
PSW4
BR4

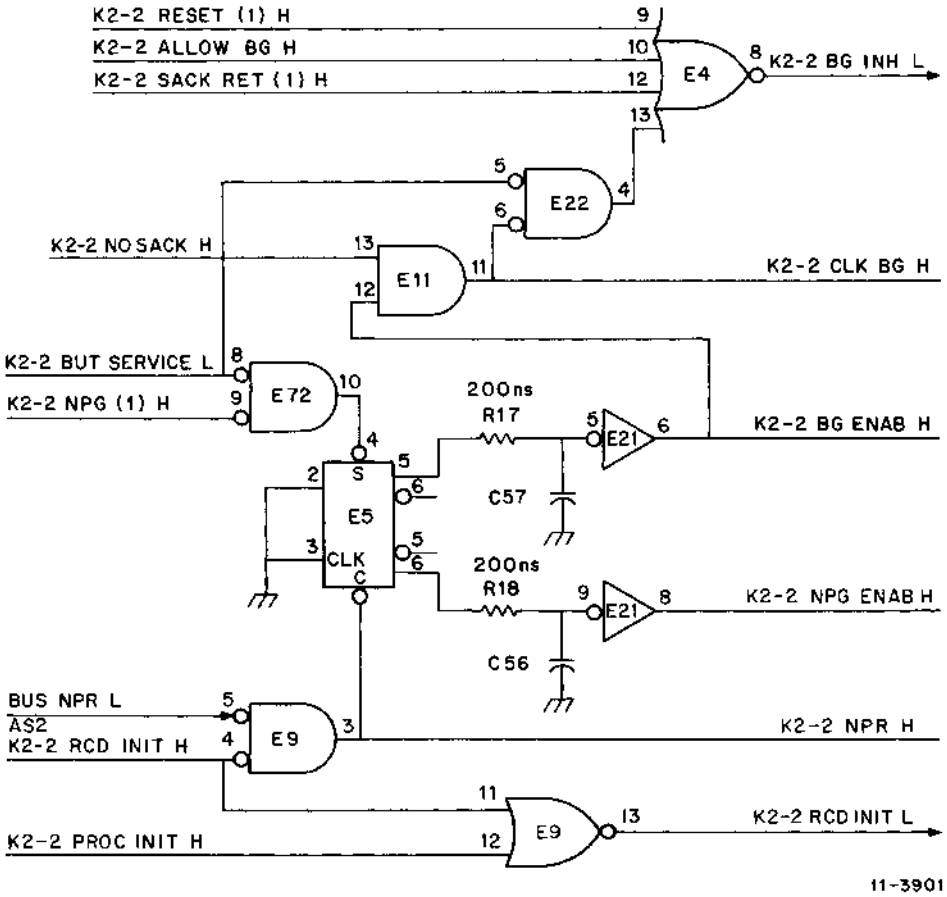


Figure 4-28 Priority Arbitration Synchronizer

If the highest BR received is of a higher priority level than the processor, the corresponding grant enable ROM output is asserted low (Figure 4-29). With no HLT RQST or trap instruction pending, the processor clock will be disabled by the K2-2 BG INH L signal. The actual bus grant is not transferred to the Unibus until the Enable BG flip-flop (E12) is set. Grants (both BG and NPG) are controlled by the synchronizer logic shown in Figure 4-28 and on print K2-2.

This circuitry arbitrates whether a bus grant (BG) or a nonprocessor grant (NPG) will result, depending on which flip-flop input line (set or reset) was deactivated first. The set input K2-2 BUT SERVICE L will cause the flip-flop to issue the BG ENAB H signal after a delay of 175 ns. Once the flip-flop is set, the bus grant arbitrated by the BR Priority Arbitration ROM (E29) is channeled onto the Unibus (bus driver E75). When the requesting peripheral receives BG, it returns BUS SACK L.

Upon receiving BUS SACK L, the processor clears its Enable BG flip-flop, removing the bus grant from the Unibus, and sets the SACK RET flip-flop to keep the processor clock disabled. Removal of bus grant causes the peripheral to drop its BUS SACK L (provided that BBSY is unasserted), assert BUS INTR L and BBSYL, and enable a vector address onto the Unibus data lines. The processor then deskews the removal of SACK, clears the SACK RET flip-flop (E5), and enables the processor clock again. Once in operation, the processor clocks the peripheral vector address into the B REG, returns BUS SSYN L, and begins running the microcode trap routine that branches the processor to the interrupt handling program determined by the vector obtained.

4.10.2 Nonprocessor Requests (NPRs)

NPRs are a facility of the Unibus that permit devices on the Unibus to communicate with each other with minimal participation of the processor. The function of the processor in servicing an NPR is to yield control of the bus in a manner that does not disturb the execution of an instruction by the processor. For example, the processor will not relinquish the bus following the DATI portion of a DATIP transfer.

When the reset input of E5 (K2-2 NPR H) becomes unasserted before the set input, and BUS SACK L is not true, the flip-flop issues K2-2 NPG ENAB H, enabling the BUS NPG H Unibus line and granting the bus to the DMA device. The requesting device then returns BUS SACK L, clearing the NPG, and waits until the bus is free (no BBSY).

4.10.3 Halt Grant Requests

The KD11-E implements what is, in effect, another priority level by monitoring the HALT/CONTINUE switch on the front panel. When a Halt is detected (HLT RQST L asserted), the processor recognizes it as an interrupt request (refer to priority levels in Paragraph 4.10.1) upon entering the next service microstate. The processor then inhibits the processor clock and returns a recognition signal (K2-2 HLT GRANT H), causing the console to drop HLT RQST L and assert BUS SACK L, gaining complete control of the Unibus and the KD11-E.

The user can maintain the processor in this inactive state (Halted) indefinitely. When the HALT switch is released, the user's console releases BUS SACK L, and the processor continues operation as if nothing had happened.

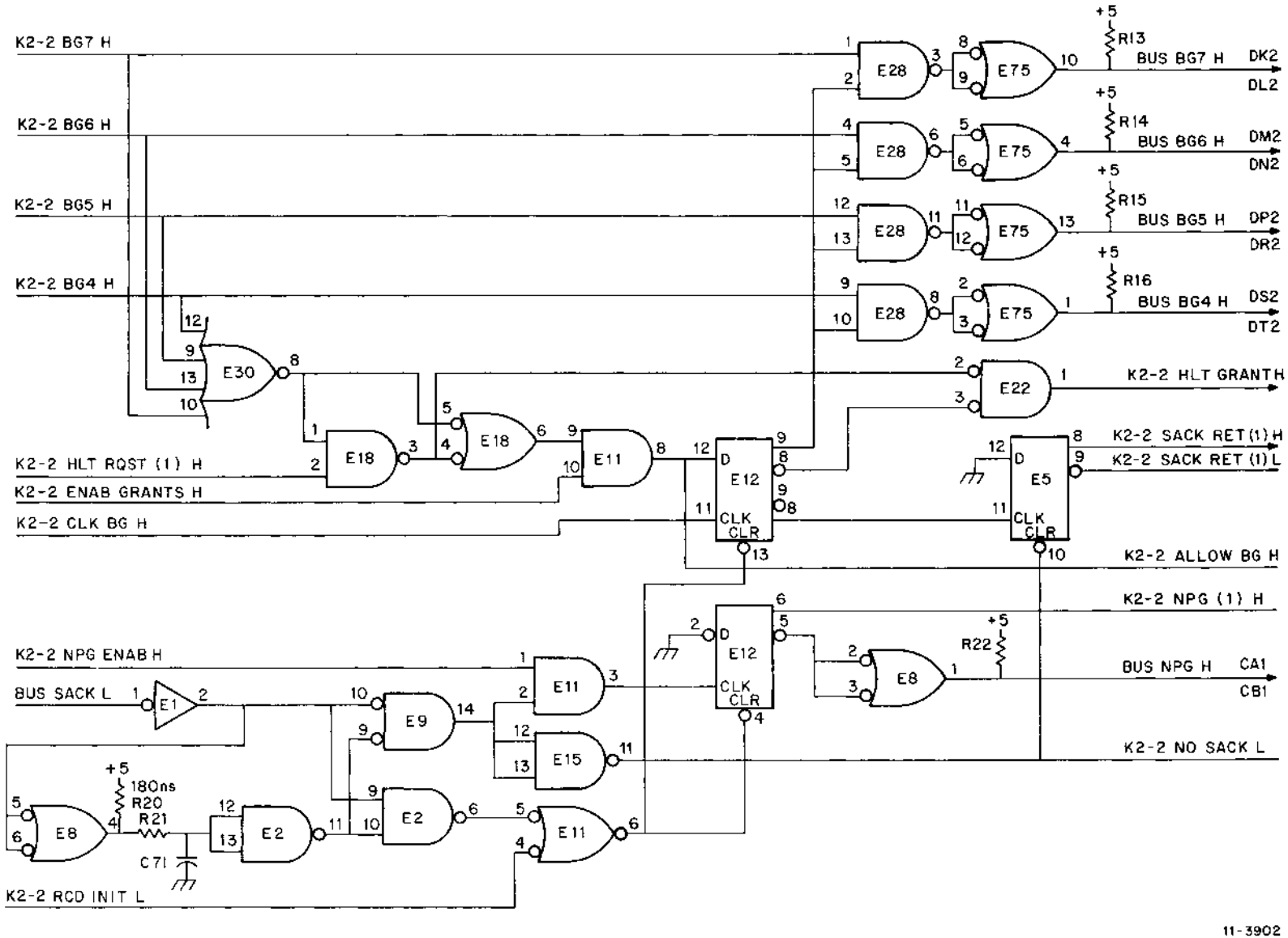


Figure 4-29 Priority Bus Control

4.11 SERVICE TRAPS

4.11.1 General Description

All interrupts, error traps, and instruction traps are recognized and serviced by the KD11-E when the processor enters what is called the service microinstruction state. The functions performed during this state are most critical to the operation of the processor.

When the service state is entered, all bus interrupts, error traps, and instruction traps realized during the performance of the last instruction are arbitrated by the service ROMs (E50 and E51 on print K2-3). Each trap condition is then serviced according to its priority, as listed in Table 4-10.

4.11.2 Circuit Operation

The service ROMs (E50 and E51 on print K2-3) service a specific trap by generating a vector address unique to that trap condition (Table 4-11). Upon leaving the service state, the processor is forced to push its present program counter (PC) and processor status word (PSW) onto its memory stack and fetch a new PC from the location specified by the vector address. A new PSW is then obtained from the next memory location after the vector. The end result of these operations is that the processor is now performing a software subroutine written by the user that could correct or indicate the occurrence of a specific error.

The various trap conditions that cause the processor to vector are as follows:

Bus Errors	A bus error indicates that the processor has attempted to access nonexistent memory or odd address (non-byte), or a memory location that did not return BUS SSYN within 22 μs. The detection circuitry for bus errors is described in Paragraph 4.7.2.6 of this manual.
Stack Overflow Error	Any attempt by the processor to decrement the contents of the Stack Pointer register (R6) below the 400-location stack limit (K1-10 8–15 = 0 H will result in the Stack Overflow flip-flop (E24 on K 2-3) being set on the next transition of K1-5 PROC CLK L. [Note that this does not apply to user stack (R16).]
Parity Error	Parity error detection circuitry is described in Paragraph 4.7.2.7.
Power Failure	The power failure circuitry is described in Paragraph 4.8.
Trace Trap	This trap is program-controlled by the user, allowing him to insert a processor/user interactive subroutine into his main program. The circuitry is described in Paragraph 4.2.6.
Reserved Instructions Illegal Instructions EMT Instructions Trap Instructions	Signals IR CODE 00 L–IR CODE 02 L are generated by the IR Decode ROMs on K2-6 for these conditions. Their decoding is discussed in Paragraph 4.5.3.

Upon entering the service microinstruction state, the service ROMs (E50 and E51 K2-3) monitor any combination of the above trap conditions which, if true, cause the assertion of microprocessor address line K2-7 MPC 00 L. While still in the service state, the ROM also generates a specific vector address (Table 4-11), using outputs K2-3 C2 H, K2-3 C3 H, and K2-3 C4 H, and channels it onto the processor AMUX lines to the SSMUX by activating K1-10 AMUX S0 H.

Table 4-11 Vector Addresses

Octal Unibus Vector Address	Trap Conditions
004	Time-Out, Odd Address, and Stack Overflow Errors
010	Illegal and Reserved Instructions
014	T-Bit Trap (BPT)
020	Input/Output Trap (IOT)
024	Power Fail
030	Emulator Trap (EMT)
034	Trap Instruction
114	Memory Parity Errors
250	Memory Management Errors

Before leaving the service state, the service ROMs also clear the condition that caused the original trap. This is done either by asserting K2-3 STOV SERV H or K2-3 PFAIL SERV H, or by performing the steps in the trap service routine. For those traps specified by the IR Code lines, however, it is necessary to remove the instruction in the IR. This is done through microcode output K2-9 BUT SERVICE (1) H, which ORs with K2-2 PROC INIT H to generate K2-3 SERV IR H and, hence, K2-3 SERV IR (1) L, removing the trap instruction from the IR. This prevents the processor from looping on the same trap condition.

For bus requests (BRs), the BUS INTR L control signal is allowed to force K2-7 MPC 00 L during service, provided that there are no other traps of higher priority. By enabling this line, the processor will branch to the trap routine. Higher priority BR interrupts are prevented from receiving BG by K2-9 BUT SERVICE (1) H.

4.12 MEMORY MANAGEMENT

4.12.1 General

4.12.1.1 Introduction - This section describes the memory management unit of the KD11-E Central Processor. The KD11-E provides the hardware facilities necessary for complete memory management and protection. It is designed to be a memory management facility for systems where the memory size is greater than 28K words and for multiuser, multiprogramming systems where protection and relocation facilities are necessary.

4.12.1.2 Programming - The memory management hardware has been optimized toward a multiprogramming environment and the processor can operate in two modes, Kernel and User. When in Kernel mode, the program has complete control and can execute all instructions. Monitors and supervisory programs would be executed in this mode.

When in User mode, the program is prevented from executing certain instructions that could:

1. Cause the modification of the Kernel program.
2. Halt the computer.
3. Use memory space assigned to the Kernel or other users.
4. Issue a Reset.

In a multiprogramming environment, several user programs could be resident in memory at any given time. The task of the supervisory program would be to: control the execution of the various user programs, manage the allocation of memory and peripheral device resources, and safeguard the integrity of the system as a whole by careful control of each user program.

In a multiprogramming system, the management unit provides the means for assigning pages (relocatable memory segments) to a user program and preventing that user from making any unauthorized access to those pages outside his assigned area. Thus, a user can effectively be prevented from accidental or willful destruction of any other user program or the system executive program.

Hardware-implemented features enable the operating system to dynamically allocate memory upon demand while a program is being run. These features are particularly useful when running higher level language programs, where, for example, arrays are constructed at execution time. No fixed space is reserved for them by the compiler. Lacking dynamic memory allocation capability, the program would have to calculate and allow sufficient memory space to accommodate the worst case. Memory management eliminates this time-consuming and wasteful procedure.

4.12.1.3 Basic Addressing – The addresses generated by all PDP-11 family central processor units (CPUs) are 18-bit addresses. Although the PDP-11 family word length is 16 bits, the Unibus and CPU addressing logic actually is 18 bits. Thus, while the PDP-11 word can only contain address references up to 32K words (64K bytes) the CPU and Unibus can reference addresses up to 128K words (256K bytes). These extra two bits of addressing logic provide the basic framework for expanding memory references.

In addition to the word length constraint on basic memory addressing space, the uppermost 4K words of address space are always reserved for Unibus I/O device registers. In a basic PDP-11 memory configuration (without management), all address references to the uppermost 4K words of 16-bit address space (160000–177777) are converted to full 18-bit references with bits 17 and 16 always set to 1. Thus, a 16-bit reference to the I/O device register at address 173224 is automatically internally converted to a full 18-bit reference to the register at address 773224. Accordingly, the basic PDP-11 configuration can directly address up to 28K words of true memory, and 4K words of Unibus I/O device registers.

4.12.1.4 Active Page Registers – The memory management unit uses two sets of eight 32-bit Active Page registers (shown on print K1-7). An APR is actually a pair of 16-bit registers: a Page Address register (PAR) and a Page Descriptor register (PDR). These registers are always used as a pair and contain all the information needed to describe and relocate the currently active memory pages (Figure 4-30).

One set of APRs is used in Kernel mode, and the other in User mode. The choice of which set to be used is determined by the current CPU mode contained in the processor status word.

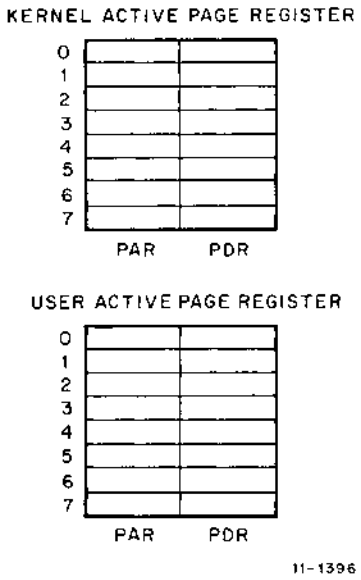


Figure 4-30 Active Page Registers

4.12.1.5 Capabilities Provided by Memory Management

Memory Size (words)	124K, max (plus 4K for I/O and registers)
Address Space	Virtual (16 bits) Physical (18 bits)
Modes of Operation	Kernel and User
Stack Pointers	2 (one for each mode)
Memory Relocation	
Number of Pages	16 (8 for each mode)
Page Length	32 to 4096 words
Memory Protection	No access Read-only Read/write

4.12.2 Relocation

4.12.2.1 Virtual Addressing – When the memory management unit is operating, the normal 16-bit direct address is no longer interpreted as a direct physical address (BA) but as a virtual address (VBA) containing information to be used in constructing a new 18-bit physical address. The information contained in the VBA is combined with relocation and description information contained in the Active Page register (APR) to yield an 18-bit BA.

Because addresses are automatically relocated, the computer may be considered to be operating in virtual address space. This means that no matter where a program is loaded into physical memory, it will not have to be "relinked"; it always appears to be at the same virtual location in memory.

The virtual address space is divided into eight 4K-word pages. Each page is relocated separately. This is a useful feature in multiprogrammed timesharing systems. It permits a new large program to be loaded into discontinuous blocks of physical memory.

A page may be as small as 32 words, so that short procedures or data areas need occupy only as much memory as required. This is a useful feature in real-time control systems that contain many separate small tasks. It is also a useful feature for stack and buffer control.

A basic function is to perform memory relocation and provide extended memory addressing capability for systems with more than 28K of physical memory. Two sets of Page Address registers are used to relocate virtual addresses to physical addresses in memory. These sets are used as hardware relocation registers that permit several users' programs, each starting at virtual address 0, to reside simultaneously in physical memory.

4.12.2.2 Program Relocation – The Page Address registers are used to determine the starting address of each relocated program memory. Figure 4-31 shows a simplified example of the relocation concept as implemented by the circuitry on print K1-6.

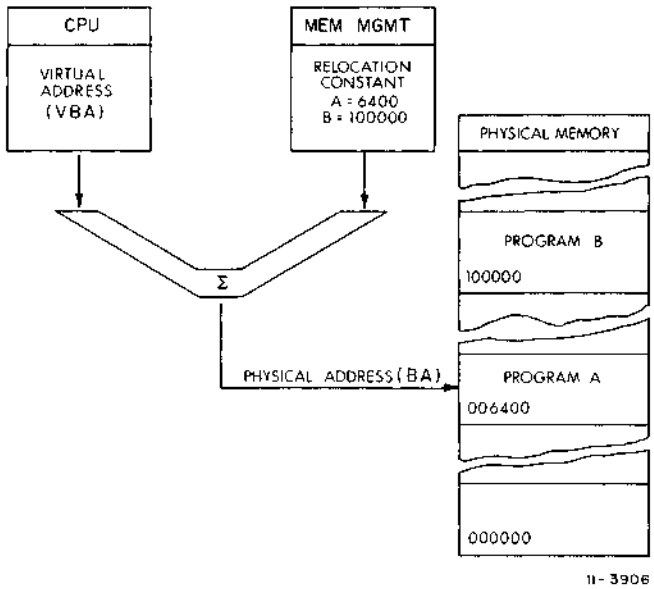


Figure 4-31 Simplified Memory Relocation Example

Program A, starting address 0, is relocated by a constant to provide physical address 6400_8. If the next processor virtual address is 2, the relocation constant will then cause physical address 6402_8, which is the second item of Program A, to be accessed. When Program B is running, the relocation constant is changed to 100000_8. Then, Program B virtual addresses, starting at 0, are relocated to access physical addresses starting at 100000_8. Using the Active Page Address registers to provide relocation eliminates the need to "relink" a program each time it is loaded into a different physical memory location. The program always appears to start at the same address.

A program is relocated in pages consisting of from 1 to 128 blocks. Each block is 32 words in length. Thus, the maximum length of a page is 4096 (128 × 32) words. Using all of the eight available Active Page registers in a set, a maximum program length of 32,768 words can be accommodated. Each of the eight pages can be relocated anywhere in the physical memory, as long as each relocated page begins on a boundary that is a multiple of 32 words. However, for pages that are smaller than 4K words, only the memory actually allocated to the page may be accessed.

The relocation example shown in Figure 4-32 illustrates several points about memory relocation.

1. Although the program appears to be in contiguous address space to the processor, the 32K-word physical address space is actually scattered through several separate areas of physical memory. As long as the total available physical memory space is adequate, a program can be loaded. The physical memory space need not be contiguous.

2. Pages may be relocated to higher or lower physical addresses with respect to their virtual address ranges. In the example shown in Figure 4-32, page 1 is relocated to a higher range of physical addresses, page 4 is relocated to a lower range, and page 3 is not relocated (even though its relocation constant is non-zero).

3. All of the pages shown in the example start on 32-word boundaries.

4. Each page is relocated independently. There is no reason why two or more pages could not be relocated to the same physical memory space. Using more than one Page Address register in the set to access the same space would be one way of providing different memory access rights to the same data, depending on which part of a program was referencing that data.

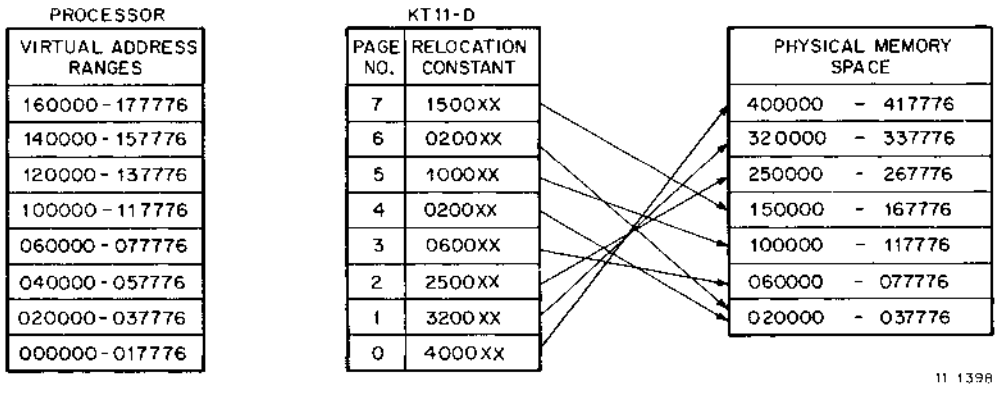


Figure 4-32 Relocation of a 32K Word Program into 124K-Word Physical Memory

4.12.2.3 Memory Units

Block	32 words
Page	1 to 128 blocks (32 to 4096 words)
No. of Pages	8 per mode
Size of Relocatable Memory	27,768 words max (8 × 4096)

4.12.3 Protection

A timesharing system performs multiprogramming; it allows several programs to reside in memory simultaneously, and to operate sequentially. Access to these programs, and the memory space they occupy, must be strictly defined and controlled. Several types of memory protection must be afforded a timesharing system. For example:

1. User programs must not be allowed to expand beyond allocated space, unless authorized by the system.

2. Users must be prevented from modifying common subroutines and algorithms that are resident for all users.

3. Users must be prevented from gaining control of or modifying the operating system software.

The memory management option provides the hardware facilities to implement all of the above types of memory protection.

4.12.3.1 Inaccessible Memory – Each page has a 2-bit access control key associated with it. The key is assigned under program control. When the key is set to 0, the page is defined as non-resident. Any attempt by a user program to access a non-resident page is prevented by an immediate abort. Using this feature to provide memory protection, only those pages associated with the current program are set to legal access keys. The access control keys of all other program pages are set to 0, which prevents illegal memory references.

4.12.3.2 Read-Only Memory – The access control key for a page can be set to 2, which allows read (fetch) memory references to the pages, but immediately aborts any attempt to write into that page. This read-only type of memory protection can be afforded to pages that contain common data, subroutines, or shared algorithms. This type of memory protection allows the access rights to a given information module to be user-independent. That is, the access right to a given information module may be varied for different users by altering the access control key.

A Page Address register in each of the sets (Kernel and User modes) may be set up to reference the same physical page in memory and each may be keyed for different access rights. For example, the User access control key might be 2 (read-only access), and the Kernel access control key might be 6 (allowing complete read/write access).

4.12.3.3 Multiple Address Space – There are two complete, separate PAR/PDR sets provided: one set for Kernel mode and one set for User mode. This affords the timesharing system with another type of memory protection capability. The mode of operation is specified by the processor status word current mode field, or previous mode field, as determined by the current instruction.

Assuming the current mode PSW bits are valid, the Active Page register sets are enabled as follows:

PSW (Bits 15,14)	PAR/PDR Set Enabled
00	Kernel mode
01, 10	Illegal (all references aborted on access)
11	User mode

Thus, a User mode program is relocated by its own PAR/PDR set, as are Kernel programs. This makes it impossible for a program running in one mode to accidently reference space allocated to another mode when the Active Page registers are set correctly. For example, a user cannot transfer to Kernel space. The Kernel mode address space may be reserved for resident system monitor functions, such as the basic input/output control routines, memory management trap handlers, and timesharing scheduling modules. By dividing the types of timesharing system programs functionally between the Kernel and User modes, a minimum amount of space control housekeeping is required as the time-shared operating system sequences from one user program to the next. For example, only the user PAR/PDR sets needs to be updated as each new user program is serviced. The two PAR/PDR sets implemented in the memory management unit are shown in Figure 4-30.

4.12.4 Active Page Registers

The memory management unit provides two sets of eight Active Page registers (APRs). Each APR consists of a Page Address register (PAR) and a Page Descriptor register (PDR). These registers are always used as a pair and contain all the information required to locate and describe the current active pages for each mode of operation. One PAR/PDR set is used in Kernel mode and the other is used in User mode. The current mode bits (or in some cases, the previous mode bits) of the processor status word determine which set will be referenced for each memory access. A program operating in one mode cannot use the PAR/PDR sets of the other mode to access memory. Thus, the two sets are a key feature in providing a fully protected environment for a timesharing multiprogramming system.

A specific processor I/O address is assigned to each PAR and PDR of each set. Table 4-12 is a complete list of address assignments.

NOTE

Unibus devices (except DMA and programmer's console) cannot access PARs or PDRs. The internal address decode logic (print K1-10) allows only the processor to access these registers.

Table 4-12 PAR/PDR Address Assignments

Kernel Active Page Registers			User Active Page Registers		
No.	**PAR**	**PDR**	**No.**	**PAR**	**PDR**
0	772340	772300	0	777640	777600
1	772342	772302	1	777642	777602
2	772344	772304	2	777644	777604
3	772346	772306	3	777646	777606
4	772350	772310	4	777650	777610
5	772352	772312	5	777652	777612
6	772354	772314	6	777654	777614
7	772356	772316	7	777656	777616

In a fully protected, multiprogramming environment, the implication is that only a program operating in the Kernel mode would be allowed to write the PAR and PDR locations for the purpose of mapping user's programs. However, there are no restraints imposed by the logic that will prevent User mode programs from writing into these registers. The option of implementing such a feature in the operating system, and thus explicitly protecting these locations from user's programs, is available to the system software designer.

4.12.4.1 Page Address Registers (PAR) – The Page Address register (PAR), shown on print K1-7 and 1 in Figure 4-33 contains the 12-bit Page Address Field (PAF) that specifies the base address of the page.

Figure 4-33 Page Address Register

Bits 15–12 are unused and reserved for possible future use.

The Page Address register may be alternatively thought of as a relocation constant, or as a base register containing a base address. Either interpretation indicates the basic function of the Page Address register (PAR) in the relocation scheme.

The Page Address register (PAR) may be regarded as either a base register containing a base address or a relocation constant. Bits are fed directly from the SSMUX to an address selected by the PAR/PDR ADRS MUX (E89 on print K1-7) when enabled by K1-10 PAR & PDR LOW L. The three scratchpad memories that comprise the PAR (E76, E77, and E78 on print K1-7) are clocked by K1-5 REG CLK H. The two associated with PAR 03:00 and PAR 07:04 are enabled by K1-10 LOAD PAR LOW L, while the other (PAR 11:08) is enabled by K1-10 LOAD PAR HIGH L. Outputs of the PARs are fed directly to the KTMUX on print K1-9, and can be channeled onto the scratchpad output lines (SP15:00) when K1-10 PAR & PDR L is asserted and K1-10 KTMUX S0 L is unasserted. This allows the contents of the registers to be accessed by a DATI or DATIP.

4.12.4.2 Page Descriptor Registers – The Page Descriptor register (PDR) comprises four scratchpad memories (E79, E86, E87, and E88 on print K1-7) and contains information regarding page expansion, page length, and access control (Figure 4-34). Bits are fed directly from the SSMUX to an address selected by the PAR/PDR ADRS MUX (E89 on print K1-7) and clocked by K1-5 REG CLK H.

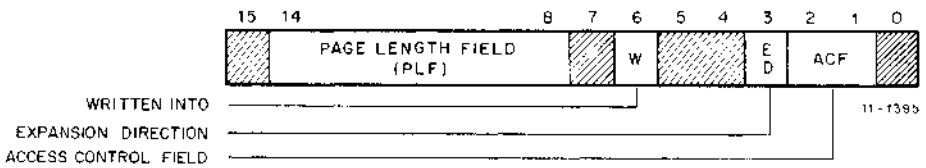


Figure 4-34 Page Descriptor Register

Access Control Field (ACF)
This 2-bit field (PDR 02:01) of the PDR describes the access rights to the specified page. The access codes or "keys" [K1-7 ACF 2 (1) H and K1-7 ACF 1 (1) H from E86] specify the manner in which a page may be accessed and whether or not a given access should result in an abort of the current operation. A memory reference that causes an abort is not completed and is terminated immediately.

Aborts are caused by attempts to access non-resident pages, page length errors, or access violations, such as attempting to write into a read-only page. All memory management traps vector through location 250 and can be used as an aid in gathering memory management information.

In the context of access control, the term "write" is used to indicate the action of any instruction which modifies the contents of any addressable word. A "write" is synonymous with what is usually called a "store" or "modify" in many computer systems. Table 4-13 lists the four ACF keys and their functions. The ACF is written into the PDR under program control.

Table 4-13 Access Control Field Keys

ACF	Key	Description	Function
00	0	Non-resident	Abort any attempt to access this non-resident page.
01	2	Resident read-only	Abort any attempt to write into this page.
10	4	Unused	Abort all accesses.
11	6	Resident read/write	Read or write allowed. No trap or abort occurs.

Expansion Direction (ED)
The ED bit located in PDR bit 3 indicates the authorized direction in which the page can expand. A logic 0 in this bit (ED = 0) indicates the page can expand upward from relative zero. A logic 1 in this bit (ED = 1) indicates the page can expand downward toward relative zero. The ED bit is written into the PDR under program control. When the expansion direction is upward (ED = 0), the page length is increased by adding blocks with higher relative addresses. Upward expansion is usually specified for program or data pages to add more program or table space. An example of page expansion upward is shown in Figure 4-35.

When the expansion direction is downward (ED = 1), the page length is increased by adding blocks with lower relative addresses. Downward expansion is specified for stack pages so that more stack space can be added. An example of page expansion downward is shown in Figure 4-36.

NOTE

To specify a block length of 42 for an upward-expandable page, write the highest authorized block number directly into the highest byte of PDR. Bit 15 is not used because the highest allowable block number is 177_8.

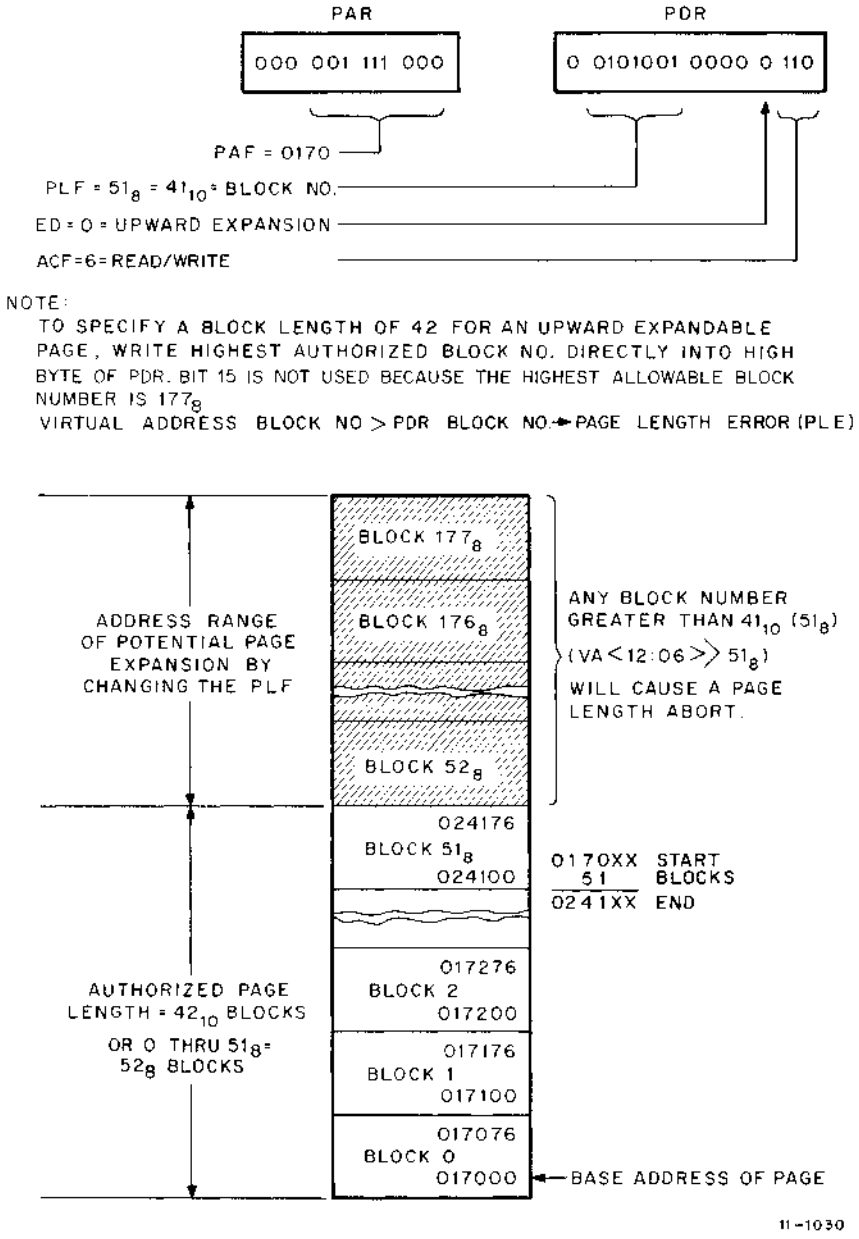
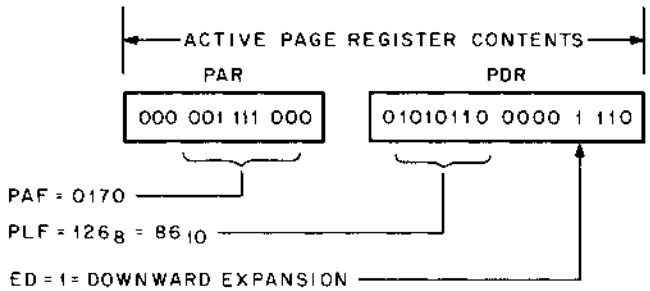


Figure 4-35 Example of an Upward-Expandable Page



TO SPECIFY PAGE LENGTH FOR A DOWNWARD EXPANDABLE PAGE, WRITE COMPLEMENT OF BLOCKS REQUIRED INTO HIGH BYTE OF PDR.

IN THIS EXAMPLE, A 42-BLOCK PAGE IS REQUIRED.
PLF IS DERIVED AS FOLLOWS:
$42_{10} = 52_8$; TWO'S COMPLEMENT = 126_8
VIRTUAL ADDRESS BLOCK NO. < PDR BLOCK NO. → PAGE LENGTH ERROR (PLE)

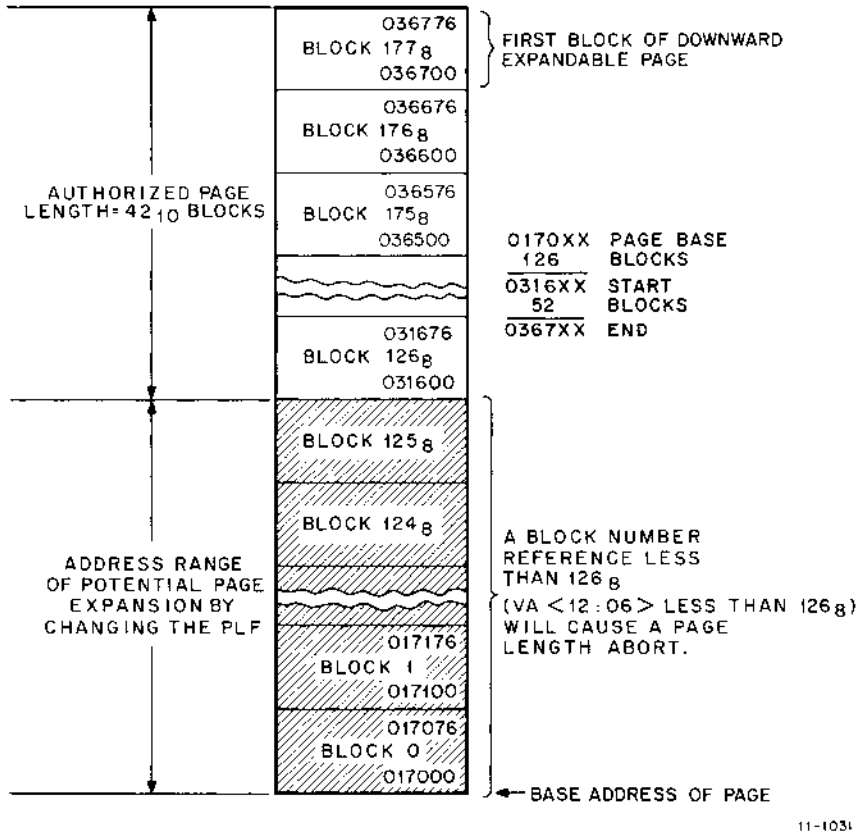


Figure 4-36 Example of a Downward-Expandable Page

Written Into (W)
The W bit located in PDR bit position 6 indicates whether the page has been written into since it was loaded into memory. W = 1 is affirmative. The W bit is automatically cleared when the PAR or PDR of that page is written into. It can only be set by the control logic (print K1-7). In disk-swapping and memory overlay applications, the W bit can be used to determine which pages in memory have been modified by a user. Those pages that have been written into must be saved in their current form; those that have not been written into (W = 1) need not be saved, and can be overlaid with new pages, if necessary.

Page Length Field
The 7-bit page length field (PLF) located in PDR bits 14:08 specifies the authorized length of the page in 32-word blocks. The PLF holds block numbers from 0 to 177_8, thus allowing any page length from 1 block to 128 blocks. The PLF is enabled by K1-10 LOAD PDR HIGH L, and written into the PDR under program control.

PLF for an Upward-Expandable Page
When the page expands upward (ED = 0), the PLF must be set to one less than the intended number of blocks authorized for that page. Thus, if the number of blocks authorized is 52_8 or 42_{10}, the PLF is set to 51_8 or 41_{10}, with block 0 being the page boundary and the first block of the page. A comparator network (E60 and E61 on print K1-8) compares the virtual address block number (VBA 12:06) with the PLF to determine whether the VBA is within the authorized page length. If the VBA block number is less than (A < B) or equal to (A = B) the PLF, the VBA is within the authorized page length. If the VBA block number is greater than (A > B) the PLF, a page length fault is detected by the hardware and K1-8 KT FAULT L is issued to the DAT TRAN circuitry on print K2-1, where it generates K2-1 ENAB ABORT H, causing a trap. When the expansion direction is upward, the page length is increased by adding blocks with higher relative addresses. Upward expansion is usually specified for program or data pages to add more program or table space (Figure 4-35).

PLF for a Downward-Expandable Page
The capability of providing downward expansion for a page is intended specifically for those pages that are to be used as stacks. In the PDP-11/34, a stack starts at the highest location reserved for it, and expands downward toward the lowest address as items are added to the stack. If the page is to be downward-expandable, the PLF must be set to authorize a page length (in blocks) that starts at the highest address of the page, which is always block 177_8. The rationale for complementing the number of blocks required to obtain the PLF is as follows:

Maximum Block No. Minus PLF ↘ ↙ Required Length Equals

$177_8 - 52_8 = 125_8$
$127_{10} - 42_{10} = 85_{10}$

Figure 4-36 contains an example of a downward-expandable page. A page length of 42 blocks is arbitrarily chosen to match the upward-expandable example shown in Figure 4-35.

NOTE
The same PAF is used in both examples. This is done to emphasize that the PAF, as the base address, always determines the lowest address of the page, whether it is upward- or downward-expandable.

4.12.5 Virtual and Physical Addresses

The memory management Unibus addressing circuitry is shown on print K1-6. When memory management is enabled (K1-8 RELOCATE H asserted), the processor ceases to load Unibus addresses directly from the scratchpad via the Bus Address register multiplexer latches (E43, E54, and E64). Instead, addresses are relocated by various constants obtained from the memory management circuitry. (Selected PAR contents are added to the VBA using adders E44, E55, and E65.)

4.12.5.1 Construction of a Physical Address – The basic information needed for the construction of a physical address (PA) comes from the virtual address (VBA), which is illustrated in Figure 4-37, and the appropriate PAR set.

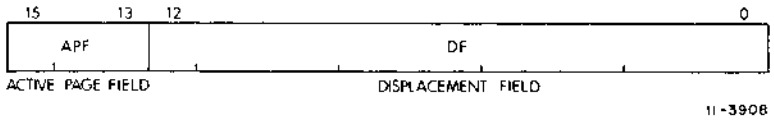


Figure 4-37 Interpretation of a Virtual Address

The virtual address (VBA) consists of:

1. The Active Page Field (APF). This 3-bit field determines which of eight Active Page registers (APR0–APR7) will be used to form the physical address (BA). The PAR/PDR ADRS MUX (E89 on print K1-7) actually selects the specific PAR.

2. The Displacement Field (DF). This 13-bit field contains an address relative to the beginning of a page. This permits page lengths up to 4K words (2^{13} = 8K bytes). The DF is further subdivided into two fields as shown in Figure 4-38.

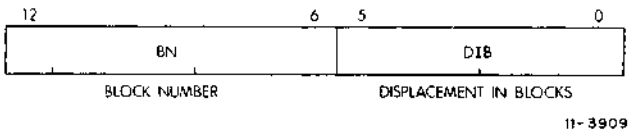


Figure 4-38 Displacement Field of Virtual Address

The displacement field (DF) consists of:

1. The Block Number (BN). This 7-bit field is interpreted as the block number within the current page.

2. The Displacement in Block (DIB). This 6-bit field contains the displacement within the block referred to by the block number.

The remainder of the information needed to construct the physical address comes from the 12-bit page address field (PAF) (part of the Active Page register) and specifies the starting address of the memory which that APR describes. The PAF is actually a block number in the physical memory, e.g., PAF = 3 indicates a starting address of 96 (3 × 32 = 96) words in physical memory.

The formation of the physical address is illustrated in Figure 4-39.

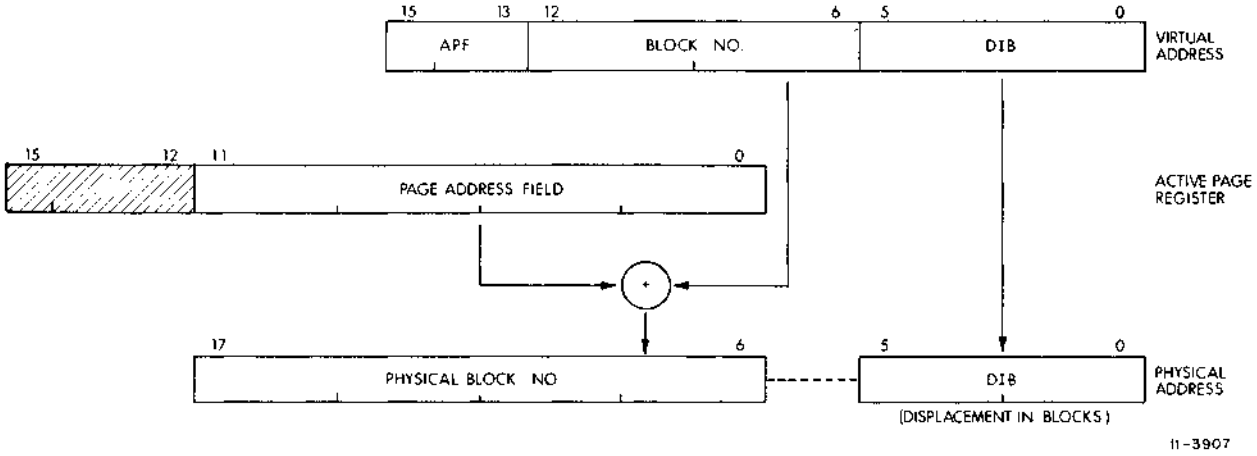


Figure 4-39 Construction of a Physical Address

The logical sequence involved in constructing a physical address is as follows:

1. Select a set of Active Page registers depending on current mode.
2. The active page field of the virtual address is used to select an Active Page register (APR0–APR7).
3. The page address field of the selected Active Page register contains the starting address of the currently active page as a block number in physical memory.
4. The block number from the virtual address is added to the block number from the page address field to yield the number of the block in physical memory which will contain the physical address being constructed.
5. The displacement in block from the displacement field of the virtual address is joined to the physical block number to yield a true 18-bit physical address.

4.12.5.2 Determining the Program Physical Address – A 16-bit virtual address can specify up to 32K words, in the range from 0 to 177776 (work boundaries are even octal numbers). The three most significant virtual address bits designate the PAR/PDR set to be referenced during page address relocation. Table 4-14 lists the virtual address ranges that specify each of the PAR/PDR sets.

Table 4-14 Relating Virtual Address to PAR/PDR Set

Virtual Address Range	PAR/PDR Set
000000–17776	0
020000–37776	1
040000–57776	2
060000–77776	3
100000–117776	4
120000–137776	5
140000–157776	6
160000–177776	7

NOTE
Any use of page lengths less than 4K words causes holes to be left in the virtual address space.

4.12.6 Status Registers

Aborts generated by the protection hardware are vectored through Kernel virtual location 250. Status registers SR0 and SR2 are used to determine why the abort occurred. Note that an abort to a location which is itself an invalid address will cause another abort. Thus the Kernel program must ensure that Kernel virtual address 250 is mapped into a valid address; otherwise a loop will occur which will require console intervention.

4.12.6.1 Status Register 0 (SR0) – SR0 contains abort error flags, memory management enable, plus other essential information required by an operating system to recover from an abort or service a memory management trap. The SR0 format is shown in Figure 4-40. Its address is 777572. Circuitry used to implement the SR0 register is shown on print K1-8.

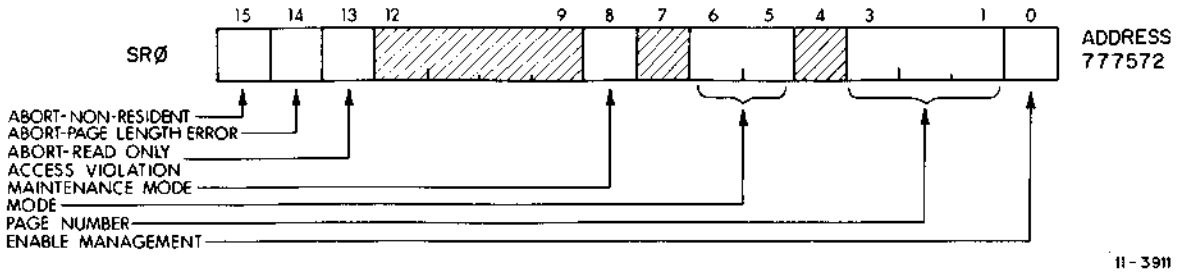


Figure 4-40 Format of Status Register 0 (SR0)

Bits 15–13 are the abort flags. They may be considered to be in a "priority queue" in that flags to the right are less significant and should be ignored. For example, a "non-resident" abort service routine would ignore page length and access control flags. A "page length" abort service routine would ignore an access control fault.

NOTE

Bit 15, 14, or 13, when set (abort conditions), causes the logic (E121 generates K1-8 ERROR H) to freeze the contents of SR0 bits 1 to 6 and status register SR2. This is done to facilitate recovery from the abort.

Protection is enabled when an address is being relocated (K1-8 RELOCATE H is active). This implies that either SR0, bit 0, is equal to 1 (memory management enabled) or that SR0, bit 8, is equal to 1 and the memory reference is the final one of a destination calculation (maintenance/destination mode).

Note that SR0 bits 0 and 8 can be set under program control to provide meaningful memory management control information. However, information written into all other bits is not meaningful. Only that information which is automatically written into these remaining bits as a result of hardware actions is useful as a monitor of the status of the memory management unit. Setting bits 15–13 under program control will not cause traps to occur. These bits, however, must be reset to 0 after an abort or trap has occurred in order to resume monitoring memory management.

Abort–Non-Resident
Bit 15 is the Abort–Non-Resident bit [K1-8 NR (1) H]. It is set by attempting to access a page with an access control field (ACF) key equal to 0 or 4 or by enabling relocation with an illegal mode in the PSW.

Abort–Page Length
Bit 14 is the Abort–Page Length bit [K1-8 PL (1) H]. It is set by attempting to access a location in a page with a block number (virtual address bits 12–6) that is outside the area authorized by the page length field (PFL) of the PDR for that page.

Abort–Read-Only
Bit 13 is the Abort–Read-Only bit [K1-8 RO (1) H]. It is set by attempting to write in a read-only page having an access key of 2.

NOTE

There are no restrictions that any abort bits could not be set simultaneously by the same access attempt.

Maintenance/Destination Mode
Bit 8 specifies maintenance use of the memory management unit. It is used for diagnostic purposes. For the instructions used in the initial diagnostic program, bit 8 is set so that only the final destination reference is relocated. It is useful to prove the capability of relocating addresses, in destination mode only.

Mode of Operation
Bits 5 and 6 indicate the CPU mode (User or Kernel) associated with the page causing the abort (Kernel = 00, User = 11).

Page Number
Bits 3-1 contain the page number of reference. Pages, like blocks, are numbered from 0 upward. The page number bit is used by the error recovery routine to identify the page being accessed if an abort occurs.

Enable Relocation and Protection
Bit 0 is the Enable bit. When it is set to 1, all addresses are relocated and protected by the memory management unit. When bit 0 is set to 0, the memory management unit is disabled and addresses are neither relocated nor protected.

4.12.6.2 Status Register 2 (SR2) – SR2 (shown on print K1-9) is loaded with the 16-bit virtual address (VBA) at the beginning of each instruction fetch but is not updated if the instruction fetch fails. SR2 is read-only; a write attempt will not modify its contents. SR2 is the virtual address program counter. Upon an abort, the result of SR0 bits 15, 14, or 13 being set will freeze SR2 until the SR0 abort flags are cleared. The address of SR2 is 777576 (Figure 4-41).

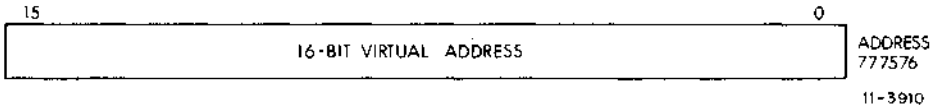


Figure 4-41 Format of Status Regiser 2 (SR2)

4.12.7 Mode Description
In Kernel mode, the operating program has unrestricted use of the machine. The program can map users' programs anywhere in core and thus explicitly protect key areas (including the device registers and the processor status word) from the user operating environment.

In User mode, a program is inhibited from executing a Halt instruction and the processor will trap through location 10 if an attempt is made to execute this instruction. A Reset instruction results in execution of a NOP (no-operation) instruction.

There are two stacks, called the Kernel stack and the User stack, used by the central processor when operating in either the Kernel or User mode, respectively.

Stack limit violations are disabled in User mode. Stack protection is provided by memory protect features.

4.12.8 Interrupt Conditions
The memory management unit relocates all addresses. Thus, when management is enabled, all trap, abort, and interrupt vectors are considered to be in Kernel mode virtual address space. When a vectored transfer occurs, control is transferred according to a new program counter (PC) and processor status word (PSW) contained in a 2-word vector relocated through the Kernel Active Page register set.

When a trap, abort, or interrupt occurs, the "push" of the old PC (old PSW) is to the User/Kernel R6 stack specified by CPU mode bits 15 (14) of the new PSW in the vector (00 = Kernel, 11 = User). The CPU mode bits also determine the new APR set. In this manner it is possible for a Kernel mode program to have complete control over service assignments for all interrupt conditions, since the interrupt vector is located in Kernel space. The Kernel program may assign the service of some of these conditions to a User mode program by simply setting the CPU mode bits of the new PSW in the vector to return control to the appropriate mode.

User Processor Status (PS) operates as follows:

PSW Bits	User RTI, RTT	User Traps, Interrupts	Explicit PSW Access
Cond. Codes (3–0)	Loaded from stack	Loaded from vector	*
Trap (4)	Loaded from stack	Loaded from vector	Cannot be changed
Priority (7–5)	Cannot be changed	Loaded from vector	*
Previous (13–12)	Cannot be changed	Copied from PS (15, 14)	*
Current (15–14)	Cannot be changed	Loaded from vector	*

*Explicit operations can be made if the processor status is mapped in user space.

4.13 CONTROL STORE

4.13.1 General Description

The Control Store circuit (prints K2-7 through K2-10) consists of twelve 512-word by 4-bit bipolar ROMs, eight hex D-type flip-flops, and an assortment of multiplexers and gates. This logic operates in a fashion similar to a microprocessor having 9 address lines and 48 data output lines with a fixed set of ROM program routines.

Each Control Store ROM location can generate a specific set of outputs capable of configuring the data path, determining the function performed by the arithmetic/logic unit (ALU), influencing the DAT TRAN circuitry, or exercising general control over the total KD11-E operation. The contents of each location are configured in such a way that sequences of locations can be combined into microroutines that perform the various PDP-11 instruction operations. Each ROM location is, therefore, considered as a microinstruction or microstep.

4.13.2 Branching Within Microroutines

Each microinstruction in the Control Store specifies the location of the next microstep in a sequence. After the execution of a microstep, the outputs of ROMs E107, E108, and E109 are latched into E89 and E91 (microprogram counter latch) to specify the location of the next microstep. Conditional branching within a microroutine is accomplished by wire-ORing signals generated by external hardware onto the MPC lines when directed by some other Control Store output. Typical wire-ORed signals include the following:

Instruction Decode — The microroutines contained in the Control Store are designed to perform efficiently the operations specified by the various PDP-11 instructions. Specific microroutines are implemented for specific instructions. The main purpose of the IR Decode circuitry is to translate the PDP-11 instruction in the IR to a set of bits that can be wire-ORed onto the MPC lines upon request (IR DECODE L), developing the next control word. A description of the specific addresses for each instruction is included in Paragraph 4.5.3 of this manual.

Trap Decode	A routine has also been included in the Micro Store to implement an error routine that pushes and pops the PC and PSW onto or off the processor stack. Upon request of the Control Store [K2-9 BUT SERVICE (1) H], the MPC 00 line can be enabled by the Service ROM (E50), causing a microbranch to this microroutine.
PWR Restart	Upon performing a power restart, the MPC is cleared by an Initialize signal (INIT). The power-up circuitry on print K2-3 then enables the MPC 00 line, forcing the Control Store to perform the power-up routine beginning at MPC address 001.

In general, microsteps are not executed from numerically sequential locations in the Control Store; therefore, care should be taken in following the flows described in Chapter 5 of this manual.

Figure 4-42 shows the format of all 512 words in the KD11-E Control Store. The fields, the possible values they contain, and the significance of each value are described below.

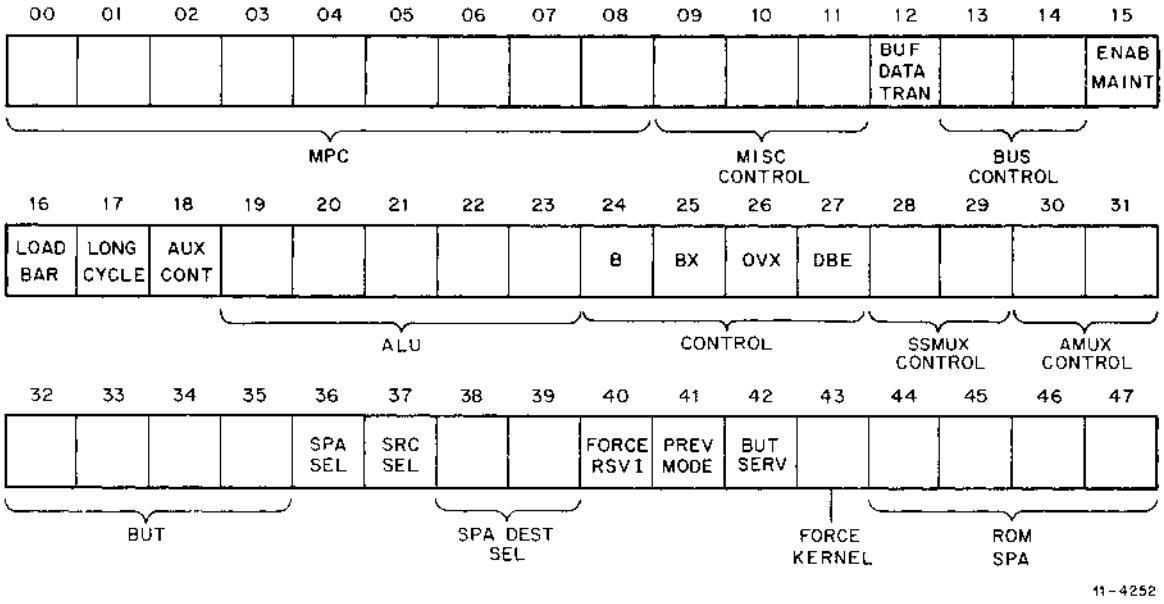


Figure 4-42 Control Store Fields

4.13.3 Control Store Fields

Use the KD11-E flow diagrams as reference for actual control field bit patterns.

Field	Field Length	Description
MPC	9	Nine-bit micro-PC address, which specifies the ROM location of the next microstep to be performed.
Miscellaneous	Control	Three multiplexed control lines that generate the following enable signals: LOAD IR L – Allows loading of the Instruction register (print K2-5). LOAD PSW L – Allows the PSW register to be loaded upon completion of this microstep (prints K1-1 through K1-4). LOAD CC L – Allows the condition codes N, Z, V, and C to be loaded upon completion of this microstep (print K1-1). BUT DEST L – Enables microbranch to destination operand microcode sequence (print K2-6). ENAB STOV L – Enables the stack overflow detection circuit (print K2-3). LOAD COUNT L – Allows the counter circuit (print K2-10) to be loaded upon completion of this microstep. CLK COUNT L – Enables the counter clock circuit (print K2-10).
BUF DAT TRAN	1	Enables the data transfer circuitry (print K2-1). Indicates that the processor is performing a Unibus transfer during this microstep.
Bus Control	2	Enables the Unibus control lines BUS C0 L and BUS C1 L, as follows:

C1 (1) H	C0 (1) H	Transfer
0	0	DATI
0	1	DATIP
1	0	DATO
1	1	DATOB

ENAB MAINT	1	Enables the memory management maintenance relocation feature.
LOAD BAR	1	Allows the Physical Bus Address register (BA on print K1-6) to be loaded during this microstep.

Field	Field Length	Description
LONG CYCLE	1	Forces the processor to perform a longer (240 ns) machine cycle during this microstep. Typically this is done during bus DATOs.
AUX CONTROL	1	Enables the Auxiliary Control ROMs during operate instruction microsteps.
ALU S3–ALU S0 ALU MODE, ALU CIN BLEG 01:00	5	Determine the operation performed by the 16-bit ALU according to Table 4-2. These lines are also wire-ORed, allowing the Auxiliary Control circuitry to determine the ALU operations according to Table 4-2.
B, BX, OVX, DBE, CONTROL	4	These multiplexed outputs control the operation of the B register and BX register during each microstep and detect overflow or double bus errors.
SSMUX CONTROL	2	Controls the select lines of the SSMUX according to the following:

Select	SS 01 H	SS 00 H
Straight	0	0
Sign Extend	0	1
Swap Bytes	1	0
External Data	1	1

AMUX CONTROL	2	Controls the select lines of the AMUX according to the following:

Data	AMUX S1	AMUX S0
PSW	0	0
ALU	0	1
Vector	1	0
Unibus	1	1

BUT BITS	4	Encoded control lines that select the specific microbranch condition that can occur during this microstep.
SPA SRC SEL	2	Controls the select lines of the scratchpad address multiplexer during the first half of this microstep.

Field Select	SEL 1	SEL 0
ROM	0	0
RS	0	1
RD	1	0
RBA	1	1

Field	Field Length	Description
SPA DST SEL	2	Controls the select lines of the scratchpad address multiplexer during the second half of this microstep.

Field Select	SEL 1	SEL 0
ROM	0	0
RS	0	1
RD	1	0
RBA	1	1

FORCE RSV1	1	Controls which source register will be selected by the scratchpad address multiplexer. If RS = is an even-numbered register, then RSV1 = Register 1. If, however, RS = an odd-numbered register, then RSV1 = the same register.
PREVIOUS MODE	1	Allows the processor to perform this microstep using the previous memory management mode [PSW (13:12)].
BUT SERVICE	1	Indicates that the processor has entered the Service microstep. Enables the Service ROM (E50), causing the processor to recognize any pending errors or interrupts.
Force Kernel	1	Forces the processor to perform this microstep in the memory management Kernel mode.
ROM SPA	4	Allows the microinstructions from the Control Store to determine which scratchpad register will be addressed during the next microstep, unless otherwise specified by the scratchpad address multiplexer control lines previously mentioned.

CHAPTER 5
MICROCODE

5.1 MICROPROGRAM FLOWS

A complete set of microinstruction flows is shown in block diagram form in the KD11-E print set. Figure 5-1 is a simplified version that provides an overview and aids in using the detailed flows. No attempt will be made in this manual to trace each path of this microcode, but the following examples should provide an adequate background for the reader.

5.2 FLOW NOTATION GLOSSARY

The block flows should be self-explanatory. To aid in understanding them, the following glossary of flow notation should be reviewed.

Designation	Definition
BA	Unibus Bus Address lines
-	Minus the operator
:	Separator
DATI	Initiate DATI operation on Unibus
+	Plus the arithmetic operator
PC	Program Counter = scratchpad register 7 (R7)
B	B register
IR	Instruction register
BX	BX register
RS	Scratchpad register specified by the source portion of the current instruction [IR (08:06)]
RD	Scratchpad register specified by the destination portion of the current instuction [IR (02:00)]
RN	Scratchpad register n specified by the Control Store ROM SPA lines
ENAB STOV	Enable the stack overflow detection logic
ENAB DBE	Enable the double bus error detection logic.
DATO	Initiate DATO operation on Unibus.
DATIP	Initiate DATIP operation on Unibus.
Rn OP B	ALU function determined by the auxiliary ALU control logic as a function of the instruction currently in the Instruction register.
BUT	Branch on microtest.
LOAD CC	Set condition codes (N, Z, V and C) according to the result of operation being performed by the ALU.
UDATA	Data being received from the Unibus data lines BUS D00 L through BUS D15 L.
RSV1	Source register specified by source portion of current instuction [IR (08:06)] ORed with a logical 1. Example: If RS is even, RSV1 would be the next highest register (RS = 4, RSV1 = 5); if, however, RS is odd, RSV1 would be the same register (RS = 5, RSV1 = 5).
←	Assignment operator.
MAINT	Indicates that the memory management Maintenance feature is enabled.
Previous	Indicates that this microstep is using the previous memory management mode.

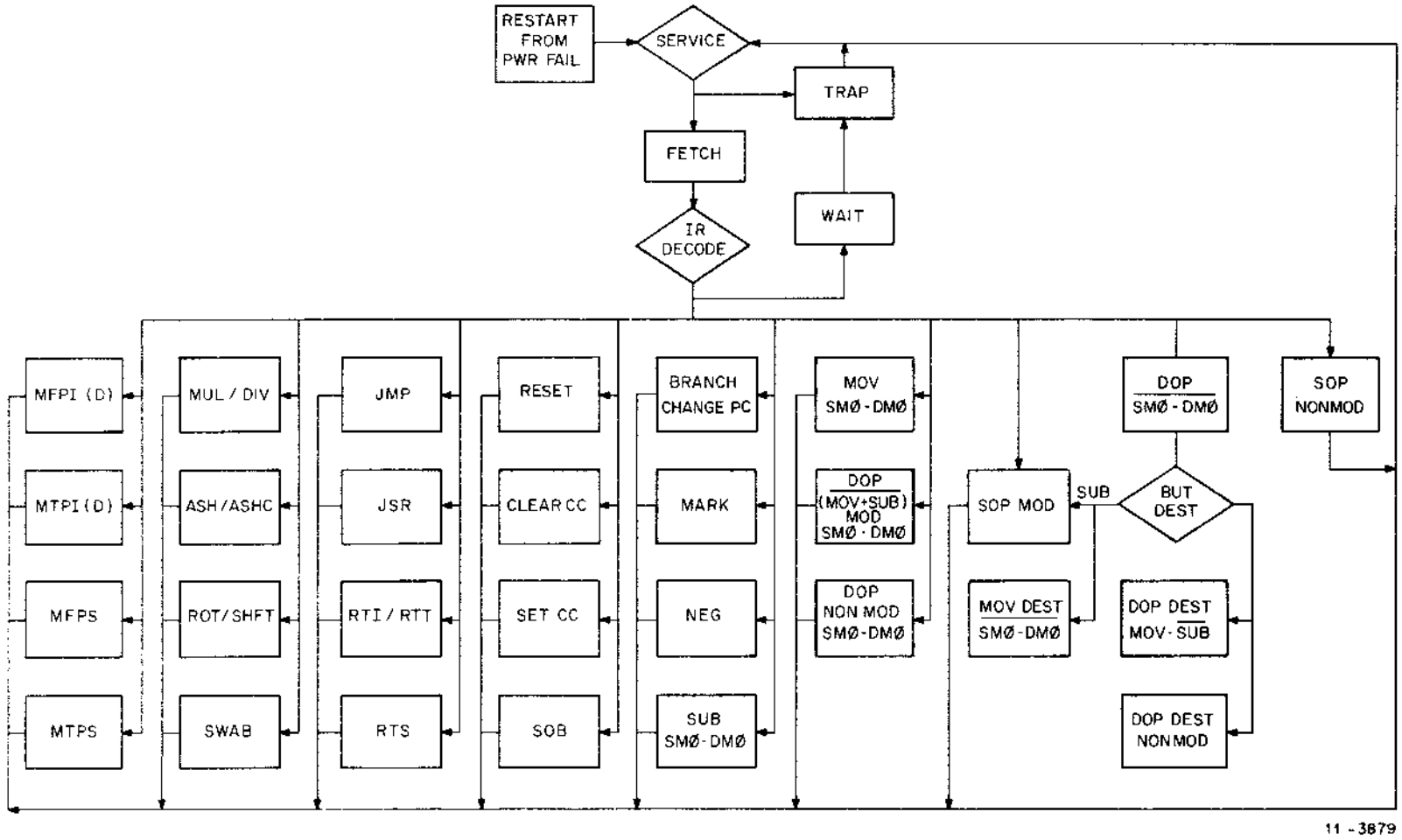


Figure 5-1 KD11-E Simplified Flow Diagram