EK-KD11D-TM-PRE

PRELIMINARY

KD11-D Processor Manual (PDP-11/04)

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

Printed in U.S.A.

Written by PDP-11 Engineering

CONTENTS

1.0 PREFACE

This manual describes the KD11D Central Processor Unit (M7263). Complete understanding of its contents requires that the user have a general knowledge of digital circuitry and a basic understanding of PDP-11 computers. The following related documents may be valuable as references.

PDP11 Peripherals Handbook
PDP11 Processor Handbooks
PDP11/04 Users Manual

2.0 OVERALL DESCRIPTION

The KD11D is a one-board central processor unit (CPU) designed for the PDP-11/04 computer series. 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 instruction decoding. It can perform data transfers directly between I/O devices and memory; does both single-and double-operand addressing and handles both 16-bit word and 8-bit byte data.

The KD11D is program compatible with the KD11B presently being used in the PDP-11/05. It also provides all the processing capability previously available at a significantly higher speed. Features available on the KD11B which are not provided on the KD11D are console, serial communication line, and line clock circuitry. These options will now be provided as separate UNIBUS options in the traditional PDP-11 sense.

3.0 INSTRUCTION SET

3.1 Introduction

The KD11D is defined by its instruction set. The sequences of processor operations are selected according to the instruction decoding. The following describes the PDP-11 instructions and instruction set addressing modes along with instruction set differences from those of the previous KD11B.

3.2 Addressing Modes

Data stored in memory must be accessed, and manipulated. Data handling is specified by a PDP-11 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/are to be used).

Since a large portion of the data handled by a computer is usually structured (in character strings, in arrays, in lists etc.) the PDP-11 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 is the address of the operand, rather than the operand itself.

3. As pointers which automatically steps through core locations. Automatically stepping forward through consecutive core locations is known as autoincrement addressing; automatically stepping backwards 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-11s also have instruction addressing mode combinations which 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 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 1 shows the formats of these two types of instructions. The addressing modes are listed in Table 1.

Figure 1 Addressing Mode Instruction Formats

3.2.1 Instruction Timing

The PDP-11 is an asynchronous processor in which, in many cases, memory and processor operations are overlapped. The execution time for an instruction is the sum of a basic instruction time and the time to determine and fetch the source and/or destination operands. Table 2 shows the addressing times required for the various mode of addressing source and destination operands. All PDP-11/04 times stated are subject to +10% variation and are based on a typical core memory access time of 375 ns, a typical MOS memory access time of 500 ns and a processor clock cycle time of 260 ns. PDP-11/05 times are based on a 310 ns processor clock cycle time and a MM11L memory.

3.3 PDP-11/04 Instructions

The PDP-11 instructions can be divided into five groupings:

1. Single-Operand Instructions (shifts, multiple precision instructions, rotates)

2. Double-Operand Instructions (arithmetic and logical instructions)

3. Program Control Instructions (branches, subroutines, traps)

4. Operate Group Instructions (processor control operations)

5. Condition Codes Operators (processor status word bit instructions)

Tables 3 through 7 list each instruction, including byte instructions for the respective instruction groups. Figure 2 shows the six different instruction formats of the instruction set, and the individual instructions in each format.

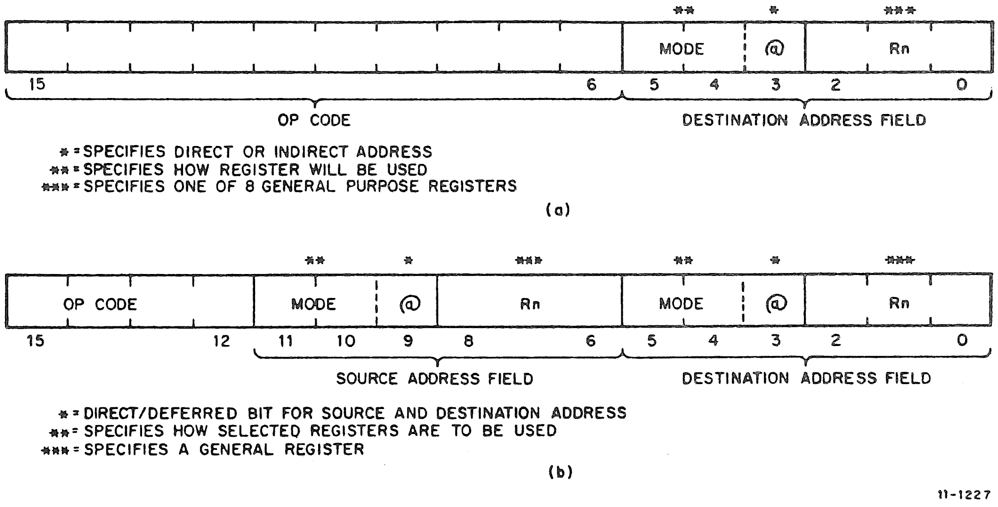

**
*
***
MODE
@
Rn
15
6
5
4
3
2
0
OP CODE
DESTINATION ADDRESS FIELD
* = SPECIFIES DIRECT OR INDIRECT ADDRESS
** = SPECIFIES HOW REGISTER WILL BE USED
*** = SPECIFIES ONE OF 8 GENERAL PURPOSE REGISTERS
(a)
OP CODE
MODE
@
Rn
MODE
@
Rn
15
12
11
10
9
8
6
5
4
3
2
0
SOURCE ADDRESS FIELD
DESTINATION ADDRESS FIELD
* = DIRECT/DEFERRED BIT FOR SOURCE AND DESTINATION ADDRESS
** = SPECIFIES HOW SELECTED REGISTERS ARE TO BE USED
*** = SPECIFIES A GENERAL REGISTER
(b)
11-1227

Figure 1 Addressing Mode Instruction Formats

3.4 Instruction Set Differences

Table 8 lists the differences between the PDP-11/05 and PDP-11/04 instruction sets.

Table 1
Addressing Modes

Binary Code	Name	Assembler Syntax	Function
			DIRECT MODES
000	Register	Rn	Register contains operand.
010	Autoincrement	(Rn)+	Register contains address of operand. Register contents incremented after reference.
100	Autodecrement	-(Rn)	Register contents decremented before reference register contains address of operand.
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) are modified.
			DEFERRED MODES
001	Register Deferred	@Rn or (Rn)	Register contains the address of the operand.
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).
101	Autodecrement	@-(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.
111	Index Deferred	@X(Rn)	Value X (stored in a 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) are modified.
			PC ADDRESSING
010	Immediate	#n	Operand follows instruction.
011	Absolute	@#A	Absolute address follows instruction.
110	Relative	A	Address of A, relative to the instruction, follows the instruction.
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)

Table 2
Basic Times

Double Operand

		Memory	Basic
Instruction	Machine	Option	Time (usec)
ADD, SUB, BIC, BIS	11/04	CORE	3.07
		CORE PARITY	3.17
		MOS	3.17
		MOS PARITY	3.33
CMP, BIT	11/04	CORE	2.81
		CORE PARITY	2.91
		MOS	2.91
		MOS PARITY	3.07
MOV	11/04	CORE	2.81
		CORE PARITY	2.91
		MOS	2.91
		MOS PARITY	3.07

Single Operand

		Memory	Basic
Instruction	Machine	Option	Time (usec)
CLR, COM, INC, DEC, NEG, ADC, SBC	11/04	CORE	2.55
		CORE PARITY	2.65
		MOS	2.65
		MOS PARITY	2.81
ROR, ROL, ASR, ASL	11/04	CORE	2.81
		CORE PARITY	2.91
		MOS	2.91
		MOS PARITY	3.07
TST	11/04	CORE	2.29
		CORE PARITY	2.39
		MOS	2.39
		MOS PARITY	2.55
SWAB	11/04	CORE	2.81
		CORE PARITY	2.91
		MOS	2.91
		MOS PARITY	3.07

Single Operand (cont'd)

		Memory	Basic
Instruction	Machine	Option	Time (usec)
All Branches (branch true)	11/04	CORE	2.55
		CORE PARITY	2.65
		MOS	2.65
		MOS PARITY	2.81
All Branches (branch false)	11/04	CORE	1.77
		CORE PARITY	1.87
		MOS	1.87
		MOS PARITY	2.03

Jump Instructions

		Memory	Basic
Instruction	Machine	Option	Time (usec)
JMP	11/04	CORE	0.84
		CORE PARITY	0.81
		MOS	0.91
		MOS PARITY	0.88
JSR	11/04	CORE	3.27
		CORE PARITY	3.27
		MOS	3.27
		MOS PARITY	3.27

Control, Trap, and Miscellaneous Instructions

		Memory	Basic
Instruction	Machine	Option	Time (usec)
RTS	11/04	CORE	3.91
		CORE PARITY	4.11
		MOS	4.11
		MOS PARITY	4.43
RTI, RTT	11/04	CORE	5.01
		CORE PARITY	5.31
		MOS	5.31
		MOS PARITY	5.79
Set N,Z,V,C	11/04	CORE	2.29
		CORE PARITY	2.39
		MOS	2.39

		MOS PARITY	2.55
Clear N,Z,V,C	11/Ø4	CORE	2.29
		CORE PARITY	2.39
		MOS	2.39
		MOS PARITY	2.55
HALT	11/Ø4	CORE	1.36
		CORE PARITY	1.46
		MOS	1.46
		MOS PARITY	1.62
WAIT	11/Ø4	CORE	2.Ø3
		CORE PARITY	2.13
		MOS	2.13
		MOS PARITY	2.29
RESET	11/Ø4	CORE	1ØØ ms
		CORE PARITY	1ØØ ms
		MOS	1ØØ ms
		MOS PARITY	1ØØ ms
IOT, EMT, TRAP, BPT	11/Ø4	CORE	7.79
		CORE PARITY	8.16
		MOS	7.95
		MOS PARITY	8.49

Table 2a.
Addressing Times

ADDRESSING FORMAT					TIME (us)	
Mode	Description	Symbolic	Machine	Memory Option	Source*	Destination**
0	REGISTER	R	11/04	CORE	0	0
				CORE PARITY	0	0
				MOS	0	0
				MOS PARITY	0	0
1	REGISTER DEFERRED	@R or (R)	11/04	CORE	0.86	1.45
				CORE PARITY	0.94	1.58
				MOS	0.94	1.48
				MOS PARITY	1.10	1.67
2	AUTOINCREMENT	(R)+	11/04	CORE	1.10	1.71
				CORE PARITY	1.20	1.84
				MOS	1.20	1.76
				MOS PARITY	1.36	1.95
3	AUTOINCREMENT DEFERRED	@(R)+	11/04	CORE	2.46	3.07
				CORE PARITY	2.66	3.30
				MOS	2.66	3.20
				MOS PARITY	2.98	3.55
4	AUTODECREMENT	-(R)	11/04	CORE	1.10	1.71
				CORE PARITY	1.20	1.84
				MOS	1.20	1.76
				MOS PARITY	1.36	1.95
5	AUTODECREMENT DEFERRED	@-(R)	11/04	CORE	2.46	3.07
				CORE PARITY	2.66	3.30
				MOS	2.66	3.20
				MOS PARITY	2.98	3.55
6	INDEXED	+X(R)	11/04	CORE	2.72	3.33
				CORE PAPITY	2.92	3.56
				MOS	2.92	3.46
				MOS PARITY	3.24	3.81
7	INDEXED	@+X(R) OR @ (R)	11/04	CORE	4.08	4.69
				COFF PARITY	4.38	5.02
				MOS	4.38	4.92
				MOS PARITY	4.86	5.43

		Machine	Memory Option	Time (us)
*For Source time, add the folowing for odd byte addressing		11/Ø4	N/A	Ø.52
**For Destination time, modify as follows:				
a.	Add for odd byte addressing with a non-modifying instruction	11/Ø4		Ø.52
b.	Add for odd byte addressing with a modifying instruction MODES 1-7	11/Ø4		1.Ø4
c.	Subtract for all non-modifying instructions except MODE Ø	11/Ø4	CORE	Ø.61
			CORE PARITY	Ø.64
			MOS	Ø.54
			MOS PARITY	Ø.57
d.	Add for MOVE instructions MODES 1-7	11/Ø4	N/A	Ø.26
e.	Subtract for JUMP and JSR instructions MODES 3, 5, 6, 7	11/Ø4		Ø.52
f.	Add for all ROTATE even byte instructions	11/Ø5 11AØ5		Ø Ø.25
g.	Add for all ROTATE odd byte instructions - Modes 1,2,4	11/Ø5 11AØ5	N/A	Ø 1.25
h.	Add for all ROTATE odd byte instructions except Modes 0,1,2,4	11/Ø5 11AØ5		Ø Ø.75

Table 3
Single Operand Instructions

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description
CLR CLRB 3.4 μs	0050DD* 1050DD	$(dst)^{\dagger} \leftarrow 0$	N: cleared Z: set V: cleared C: cleared	Contents of specified destination are replaced with zeroes.
COM COMB 3.4 μs	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 3.4 μs	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 effected	Add 1 to the contents of the destination.
DEC DECB 3.4 μs	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 effected	Subtract 1 from the contents of the destination.
NEG NEGB 3.4 μs	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 3.4 μs	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 result.

Table 3 (Cont)
Single Operand Instructions

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description
SBC SBCB 3.4 μs	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 3.4 μs	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 3.4 μs	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 3.4 μs	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 3 (Cont)
Single Operand Instructions

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description
ASR ASRB 3.4 μs	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 3.4 μs	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 3 (Cont)
Single Operand Instructions

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description
JMP 1.0 µs	0001DD	PC ← (dst)	Not effected.	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.
SWAB 4.3 µs	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).

* DD = destination (address mode and register)

† (dst) = destination contents

Table 4
Double Operand Instructions

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description
MOV MOVB 3.7 μs 3.1 μs mode 0	01SSDD* 11SSDD	(dst) ← (src) †	N: set if (src) < 0; cleared otherwise Z: set if (src) = 0; cleared otherwise V: cleared C: not effected	Word: Moves the source operand to the destination location. The previous contents of the destination are lost. The source operand is not effected. 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 CMPB 3.7 μs	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 uneffected. 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 4 (Cont)
Double Operand Instructions

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description
BIT BITB 3.7 μs	03SSDD 13SSDD	$(\mathrm{src}) \wedge (\mathrm{dst})$	N: set if high order bit of result set; cleared otherwise Z: set if result = 0; cleared otherwise V: cleared C: not effected	Performs logical AND comparison of the source and destination operands and modifies condition codes accordingly. Neither the source nor destination operands are effected. 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 3.7 μs	04SSDD 14SSDD	$(\mathrm{dst}) \leftarrow \sim (\mathrm{src}) \wedge (\mathrm{dst})$	N: set if high order bit of result set, cleared otherwise Z: set if result = 0; cleared otherwise V: cleared C: not effected	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 uneffected.
BIS BISB 3.7 μs	05SSDD 15SSDD	$(\mathrm{dst}) \leftarrow (\mathrm{src}) \wedge (\mathrm{dst})$	N: set if high order bit of result set; cleared otherwise Z: set if result = 0; cleared otherwise V: cleared C: not effected	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	06SSDD	$(\mathrm{dst}) \leftarrow (\mathrm{src}) + (\mathrm{dst})$	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 effected. Two's complement addition is performed.

Table 4 (Cont)
Double Operand Instructions

Mnemonic/ Instruction Time	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 3.7 μs	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 effected. In double precision arithmetic, the C-bit, when set, indicates a borrow.

* SS = source (address mode and register)

† (src) = source contents

Table 5
Program Control Instructions

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description
BR 2.5 μs	000400 xxx†	PC ← PC + (2 X offset)	Uneffected	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
1.9 μs no branch
2.5 μs branch
	001000
	xxx
		PC ← PC +
		(2 X offset)
		if Z = 0
			Uneffected	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
1.9 μs no branch
2.5 μs branch
	001400
	xxx
		PC ← PC +
		(2 X offset) if
		Z = 1
			Uneffected	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 1.9 μs no branch 2.5 μs branch	002000 xxx	PC ← PC + (2 X offset) if N ∨ V = 0	Uneffected	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 5 (Cont)
Program Control Instructions

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description

BHI
1.9 μs no branch
2.5 μs branch
	101000
	xxx
		PC ← PC +
		(2 X offset) if
		C = 0
			Uneffected	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 1.9 μs no branch 2.5 μs branch	101400 xxx	PC ← PC + (2 X offset) if C v Z = 1	Uneffected	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
1.9 μs no branch
2.5 μs branch
	102000
	xxx
		PC ← PC +
		(2 X offset) if
		V = 0
			Uneffected	Tests the state of the V-bit and causes a branch if the V-bit is clear. BVC is complementary operation to BVS.

BVS
1.9 μs no branch
2.5 μs branch
	102400
	xxx
		PC ← PC +
		(2 X offset) if
		V = 1
			Uneffected	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
1.9 μs no branch
2.5 μs branch
	103000
	xxx
		PC ← PC +
		(2 X offset) if
		C = 0
			Uneffected	Tests the state of the C-bit and causes a branch if C is clear. BCC is the complementary operation to BCS.

BCS
BLO
1.9 μs no branch
2.5 μs branch
	103400
	xxx
		PC ← PC +
		(2 X offset) if
		C = 1
			Uneffected	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 5 (Cont)
Program Control Instructions

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description
BLT 1.9 μs no branch 2.5 μs branch	002400 xxx	PC ← PC + (2 X offset) if N V = 1	Uneffected	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 1.9 μs no branch 2.5 μs branch	003000 xxx	PC ← PC + (2 X offset) if Z v (N $\veebar$ V) = 0	Uneffected	Operation of BGT is similar to BGE, except BGT does not cause a branch on a 0 result.
BLE 1.9 μs no branch 2.5 μs branch	003400 xxx	PC ← PC + (2 X offset) if Z v (N $\veebar$ V) = 1	Uneffected	Operation is similar to BLT but in addition will cause a branch if the result of the previous operation was 0.
BPL 1.9 μs no branch 2.5 μs branch	100000 xxx	PC ← PC + (2 X offset) if N = 0	Uneffected	Tests the state of the N-bit and causes a branch if N is clear. BPL is the complementary operation of BMI.
BMI 1.9 μs no branch 2.5 μs branch	100400 xxx	PC ← PC + (2 X offset) if N = 1	Uneffected	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 5 (Cont)
Program Control Instructions

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description

(No mnemonic)
8.2 μs
	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
8.2 μs
	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.

EMT
8.2 μs
	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
8.2 μs
	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 uneffected by these instructions

†xxx = offset, 8 bits (0-7) of instruction format
R = register (linkage pointer)

Table. 5 (Cont)
Program Control Instruction

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description
JRS 3.8 μs	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)	Uneffected	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 3.8 μs	00020R	PC ← (reg) (reg) ← SP ↑	Uneffected	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 6
Operate Group Instructions

Mnemonic/ Instruction Time	OP Code	Operation	Condition Codes	Description
HALT 1.8 μs	000000		Not effected	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.
WAIT 1.8 μs	000001		Not effected	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, since 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 20 ms	000005	PC (SP) PSW (SP)	Not effected	Sends INIT on the Unibus for 20 ms. All devices on the Unibus are reset to their state at power-up.

Table 7
Condition Code Operators

Mnemonic/ Instruction Time	OP Code	Description
CLC	000241	Set and clear condition code bits. Selectable combination of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bit 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., set the bit specified by bit 0, 1, 2, or 3 if bit 4 as a 1. Clear corresponding bits if bit 4=0.
CLZ	000242	
CLN	000244	
CLV	000250	
Set all CCs	000277	
Clear all CCs	000257	
Clear V and C		
No operation		
No operation		
	000240	
	000260	

Figure 2 PDP-11 Instruction Formats

Page 28A

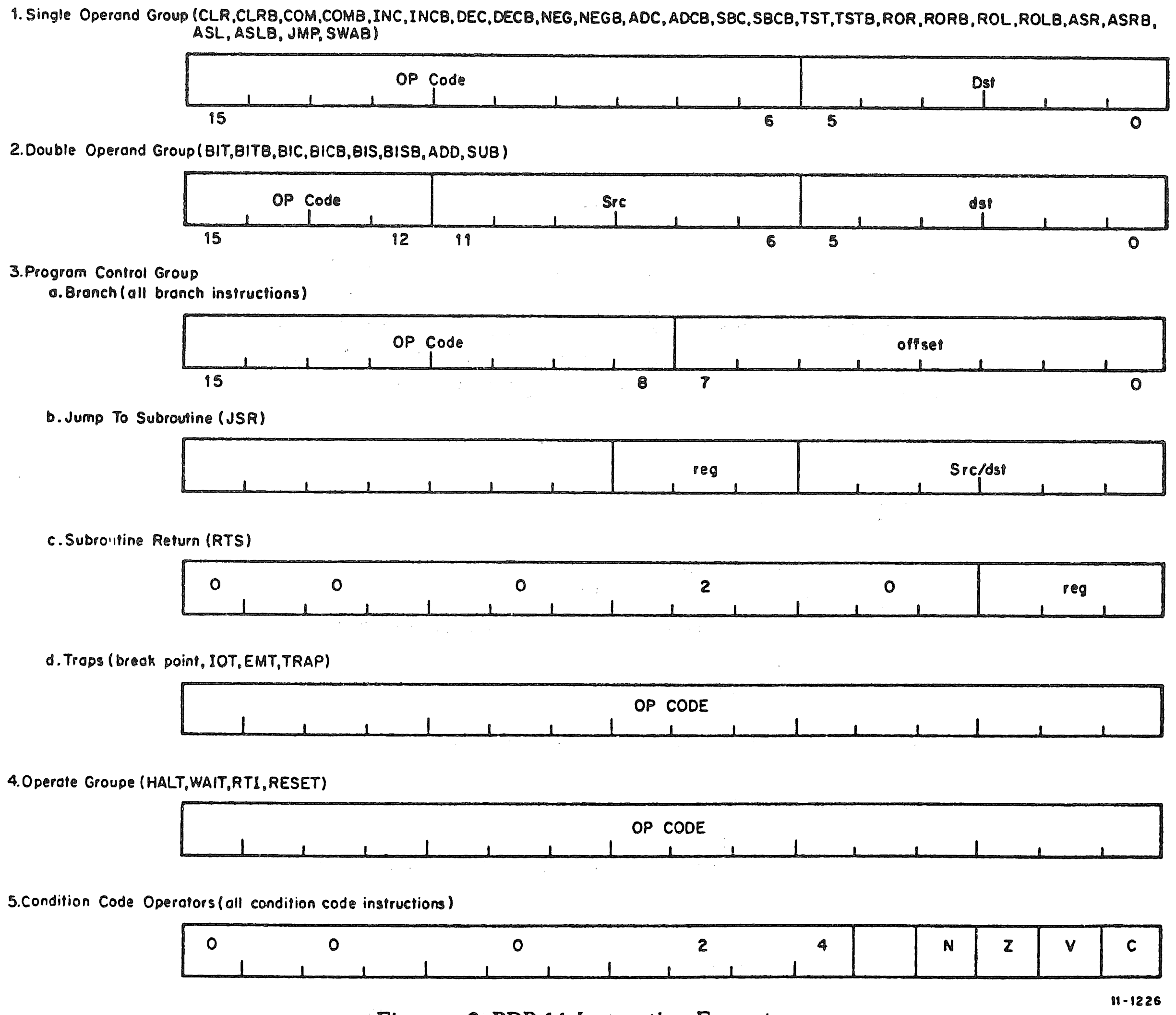

1. Single Operand Group (CLR,CLRB,COM,COMB,INC,INCB,DEC,DECB,NEG,NEGB,ADC,ADCB,SBC,SBCB,TST,TSTB,ROR,RORB,ROL,ROLB,ASR,ASRB, ASL, ASLB, JMP, SWAB)
OP Code
Dst
15
6
5
0
2. Double Operand Group (BIT,BITB,BIC,BICB,BIS,BISB,ADD,SUB)
OP Code
Src
dst
15
12
11
6
5
0
3. Program Control Group
a. Branch (all branch instructions)
OP Code
offset
15
8
7
0
b. Jump To Subroutine (JSR)
reg
Src/dst
c. Subroutine Return (RTS)
0
0
0
2
0
reg
d. Traps (break point, IOT, EMT, TRAP)
OP CODE
4. Operate Groupe (HALT, WAIT, RTI, RESET)
OP CODE
5. Condition Code Operators (all condition code instructions)
0
0
0
2
4
N
Z
V
C
11-1226

Figure 2 PDP-11 Instruction Formats

TABLE 8
DIFFERENCES:

PDP-11/05 (KD11B)

I. If a BUS ERROR occurs due to a transfer to nonexistent memory during an autoincrement transfer (Mode 2), an instruction fetch, or stack pop, the associated register will be incremented.

Examples:

a. FETCH ROUTINE

```
BA <--- PC, DATI
PC <--- PC Plus 2
(BUS ERROR detected and recognized)

BRANCH TO SERVICE
```

b. AUTOINCREMENT ROUTINE (SOURCE)

```
BA <--- R [S] , DATI, ALBYT
B  <--- R [S]  Plus BYTE BAR Plus 1
R  [S] <--- B
(BUS ERROR detected and recognized)

BRANCH TO SERVICE
```

c. AUTOINCREMENT ROUTINE (DESTINATION)

```
BA <--- R [D] , DATIP, ALBYT
B  <--- R [D]  Plus 1 Plus BYTE BAR
R  [D] <-- B
(BUS ERROR detected and recognized)

BRANCH TO SERVICE
```

d. STACK POPS

```
BA <--- R [6] , DATI
B  <--- R [6]  Plus 2
R  [6] <--- B
(BUS ERROR detected and recognized)

BRANCH TO SERVICE
```

PDP-11/04 (KD11D)

I. If a BUS ERROR occurs due to a transfer to nonexistent memory during an autoincrement transfer (Mode 2), an instruction fetch, or stack pop, the associated register will not be incremented.

Examples:

a. FETCH ROUTINE

```
BA <--- PC, DATI
B,IR <--- UNIBUS DATA
(BUS ERROR detected and recognized)

BRANCH TO SERVICE
```

b. AUTOINCREMENT ROUTINE (SOURCE)

```
BA <--- R [S] , DATI, ALBYT
B  <--- UNIBUS DATA
(BUS ERROR detected and recognized)

BRANCH TO SERVICE
```

c. AUTOINCREMENT ROUTINE (DESTINATION)

```
BA <--- R [D] , DATIP, ALBYT
B  <--- UNIBUS DATA
(BUS ERROR detected and recognized)

BRANCH TO SERVICE
```

d. STACK POPS

```
BA <--- R [6] , DATI
B  <--- UNIBUS DATA
(BUS ERROR detected and recognized)

BRANCH TO SERVICE
```

II. For JUMP autoincrement (Mode 2) instructions, JMP (R)+ or JSR reg, (R)+, the contents of R are incremented by 2, then used as the new PC address. This feature is compatible with the PDP-11/20.

III. The processor can access its general registers using their UNIBUS addresses.

```
Examples:  CLR @# 177700
           MOV R1, @# 177700
```

Note: These accesses do not operate correctly and are not supported.

IV. The KD11B CPU contains a line clock, serial communication line, and programmers console circuitry. These devices answer to the UNIBUS addresses 172540, 177560, and 177570 respectively.

V. The KD11B provides for a 15 usec SACK time-out when recognizing BUS interrupts.

VI. The console HALT switch has the lowest priority level. This feature allows the user to single-instruction-step through an interrupt.

The PDP-11/05 priority order for traps and interrupts is as follows:

BUS ERRORS
HALT INSTRUCTION
TRAP INSTRUCTIONS
TRACE TRAP
STACK OVERFLOW
POWER FAIL
INTERRUPTS
HALT ON CONSOLE

II. For JUMP autoincrement (Mode 2) instructions, JMP (R)+ or JSR reg, (R)+ the initial contents of R register are used as the new PC. This feature is compatible with the PDP-11/40.

III. The processor cannot access its general registers using their UNIBUS addresses. In the 11/04, these accesses to the general registers will return SSYN and not time out. Reads will return zero and writes will not cause any change in the register contents.

IV. The KD11D CPU does not contain a line clock, serial communication line, or console circuitry. These features will be provided as UNIBUS options in the traditional PDP-11 sense. In systems that do not contain these options, attempts to address them will result in a non-existent memory trap.

V. The KD11D has no provisions for SACK time-out on the basic CPU module. A SACK return circuit is provided on the M9302 Terminator module which must be used with the KD11D at the end of the UNIBUS. This device automatically returns SACK if no peripheral accepts a GRANT issued by the CPU.

VI. The console HALT switch has a higher priority level than interrupts and therefore, does not allow the user to single-instruction-step through an interrupt.

The PDP-11/04 priority order for traps and interrupts is as follows:

HALT INSTRUCTION
BUS ERRORS
TRAP INSTRUCTIONS
TRACE TRAP
STACK OVERFLOW
POWER FAIL
HALT SWITCH
INTERRUPTS

VII.	The KD11B has no PARITY ERROR detection capabilities and therefore does not support parity memory.	VII.	The KD11D CPU contains PARITY ERROR detection circuitry, and will support parity memory options.
VIII.	First instruction after RTI is guaranteed to be executed.	VIII.	If The RTI sets the T bit, the T bit trap is acknowledged immediately after the RTI instructions.
IX.	RTT instructions are not implemented.	IX.	First instruction after RTT is guaranteed to be executed.

TABLE OF PROGRAMMING DIFFERENCES

		11/15 & 11/20	11/05 & 11/10	11/35 & 11/40	11/04
I.	GENERAL REGISTERS (Including PC & SP)				
A.	OPR%R,(R)+ or OPR%R,-(R) OPR%R,@(R)+ OPR%R,@-(R) (Using the same reg. as both source & destination).	Contents of R are incremented by 2 (or decremented by 2) before being used as the source operand.	Initial contents of R are used as the source operand.	Same as 11/20	Same as 11/05
B.	JMP(R)+ or JSR reg, (R)+ (Jump using auto-increment mode).	Contents of R are incremented by 2, then used as the new PC address.	Same as 11/20	Initial contents of R are used as the new PC.	Same as 11/40
C.	MOV PC,@#A or MOV PC,A (Moving the incremented PC to a memory address referenced by the PC).	Location A will contain the PC of the Move instruction +4.	Location A will contain PC+2.	Same as 11/20	Same as 11/05
D.	Stack Pointer (SP), R6 used for referencing.	Using the SP for pointing to odd addresses or non-existent memory causes a HALT (double bus error).	Same as 11/20	Odd address of non-existent memory references with SP cause a fatal trap, with a new stack created at locations 0 & 2.	Same as 11/05

TABLE OF PROGRAMMING DIFFERENCES (Cont)

	11/15 & 11/20	11/05 & 11/10	11/35 & 11/40	11/04
E. Stack Overflow	Stack limit fixed at 400 (octal). Overflow (going lower) checked after @-(R6), JSR, traps, and address modes 4 & 5. Overflow serviced by an overflow trap. No red zone.	Same as 11/20	Variable limit with Stack Limit option. Overflow checked after JSR, traps, and address modes 1, 2, 4, & 6. Non-altering references to stack data is always allowed. There is a 16-word yellow (warning) zone. Red zone trap occurs if stack is 16 words below boundary; PS & PC are saved at locations 0 & 2.	Same as 11/05
II. TRAPS & INTERRUPTS				
A. RTI instruction	First instruction after RTI is guaranteed to be executed.	Same as 11/20	If RTI sets the T bit, the T bit trap is acknowledged immediately after the RTI instruction.	If RTI sets the T bit, the T bit trap is acknowledged immediately after th RTI instruction.
B. RTT instruction	Not Implemented	Not Implemented	First instruction after RTT is guaranteed to be executed. Acts like RTI on the 11/20.	First instruction after RTT is guaranteed to be executed.

TABLE OF PROGRAMMING DIFFERENCES (Cont)

		11/15 & 11/20	11/05 & 11/10	11/35 & 11/40	11/04.
C.	Processor Status (PS) odd byte at location 777-777.	Addressing odd byte of PS (bits 15-8) causes an odd address trap.	Odd byte of PS can be addressed without a trap.	Same as 11/05	Same as 11/05
D.	T bits of PS	T bit can be loaded by direct address of PS, or from the console.	Same as 11/20	Only RTI, RTT traps, and interrupts can load the T bit.	Same as 11/05
E.	Bus Errors				
1.	PC contains odd address	PC Unincremented	Same as 11/20	Same as 11/20	Same as 11/05
2.	PC contains address in nonexistent memory	PC Incremented	PC Incremented	PC Unincremented	Same as 11/05
3.	Register contains odd address and instruction Mode 2.	Register Unincremented	Same as 11/20	Register Incremented	Same as 11/05
4.	Register contains address in nonexistent memory and instruction Mode 2.	Register Incremented	Same ass 11/20	Register Incremented	Register Unincremented

TABLE OF PROGRAMMING DIFFERENCES (Cont)

	11/15 & 11/20	11/05 & 11/10	11/35 & 11/40	11/04
F. Interrupt Service Routine	The first instruction in the routine is guaranteed to be executed.	The first instruction will not be executed if another interrupt occurs at a higher priority.	Same as 11/05	Same as 11/05
G. Priority order of Traps & Interrupts	Odd address Timeout HALT from console Trap instructions Trace trap Stack overflow Power fail	Odd address Timeout HALT instruction Trap instructions Trace trap Stack overflow Power fail HALT from console	Odd address Stack overflow (red) Timeout Mem. Mgt. violation HALT Trap instructions Trace trap Stack overflow (yellow) Power fail	HALT instruction Bus errors TRAP instructions TRACE TRAP STACK overflow Power Fail HALT Switch Interrupts
III. MISCELLANEOUS				
A. SWAB and V bit	SWAB instruction conditionally sets the V bit.	V bit is cleared.	Same as 11/05	Same as 11/05 except that RTT is implemente
B. Instruction Set	Basic set.	Same as 11/20	Basic set + MARK, RTT, SOB, SXT, XOR. EIS adds: MUL, DIV, ASH, ASHC. Floating Point adds: FADD, FSUB, FMUL, FDIV.	Same as 11/05

3.5 Bus Latency Times

The typical bus latency times for bus requests (BR4 through BR7) and Non-Processor requests (NPR) are as follows. Note that there are many gaps in these timing sequences that are a function of the length and loading of the UNIBUS and the speed realized by the users peripheral circuitry.

BUS REQUESTS -

BR to BG - maximum time is length of longest instruction cycle of KD11D.

BG to SACK - function of UNIBUS length and loading and the users peripheral.

SACK to BG OFF - 300 ns

BG OFF to SACK OFF - function of peripheral.

SACK OFF to FIRST INTR ROUTINE FETCH -

Memory	Core:	6.36 us
	Core Parity	6.63
	MOS	6.43
	MOS Parity	6.81

NON-PROCESSOR REQUESTS -

NPR to NPG - 340 ns

NPR to BUS CONTROL -	Core:	3.52 us
	Core Parity	3.66
	MOS	3.56
	MOS Parity	3.75

4.0 CPU OPERATING SPECIFICATIONS

Temperature: TBD

Relative Humidity: 20% to 95% (with condensation)

Input Power: +5VDC +5% at 4 amperes.

Physical Size: Single Hex module (8 1/2 x 15 inches)

Interface Requirements: All I/O signals are available on connectors A and B. These signals are pin compatible with UNIBUS pinout as shown in Table 9.

Power and Ground Pinouts:	+5V: pins AA2, BA2, CA2, DA2, EA2, FA2. GND: pins AB2, AC2, AN1, AP1, AR1, AS1, AT1, AV2, BB2, BC2, BD1, BE1, BT1, BV2, CC2, CT1, DC2, DT1, EC2, ET1, FC2, FT1.
Number of Integrated Circuits:	137

TABLE 9
MODIFIED UNIBUS PIN ASSIGNMENTS

PIN	SIGNAL	PIN	SIGNAL
AA1	INIT L	BA1	SPARE
AA2	POWER (+5V)	BA2	POWER (+5V)
AB1	INTR L	BB1	SPARE
AB2	TEST POINT	BB2	TEST POINT
AC1	DOO L	BC1	BR 5L
AC2	GROUND	BC2	GROUND
AD1	DO2 L	BD1	BAT-BACKUP +5V
AD2	DO1 L	BD2	BR-4L
AE1	DC4 L	BE1	INT. SSYN.
AE2	DO3 L	BE2	PAR: DET.
AF1	DO6 L	BF1	ACLO L
AF2	DO5 L	BF2	DCLO L
AH1	DO8 L	BH1	AØ1 L
AH2	DO7 L	BH2	AØØ L
AJ1	D1Ø L	BJ1	AØ3 L
AJ2	DØ9 L	BJ2	AØ2 L
AK1	D12 L	BK1	AØ5 L
AK2	D11 L	BK2	AØ4 L
AL1	D14 L	BL1	AØ7 L
AL2	D13 L	BL2	AØ6 L
AM1	PA L	BM1	AØ9 L
AM2	D15 L(3)	BM2	AØ8 L
AN1	P1	BN1	A11 L
AN2	PB L	BN2	A1Ø L
AP1	PØ	BP1	A13 L
AP2	BBSY L	BP2	A12 L
AR1	BAT BACKUP +15V	BR1	A15 L
AR2	SACK L	BR2	A14 L
AS1	BAT BACKUP -15V	BS1	A17 L
AS2	NPR L	BS2	A16 L
AT1	GROUND	BT1	GROUND
AT2	BR .7L	BT2	C1 L
AU1	+2ØV	BU1	SSYN L
AU2	BR 6L	BU2	CØ 1
AV1	+2ØV	BV1	MSYN L
AV2	+2ØV	BV2	-5V

5.0 DETAILED HARDWARE DESCRIPTION

5.1 Introduction

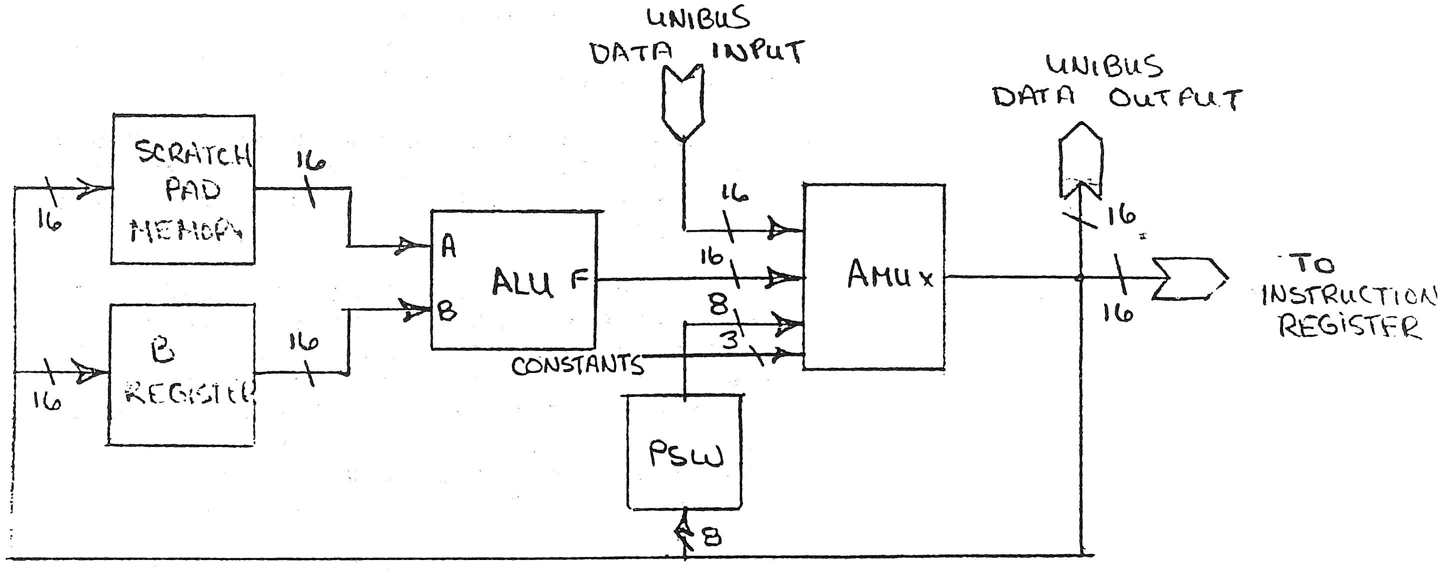
The following is a detailed circuit description of the KD11D Central Processor Unit (CPU) being used in the PDP11/04 computer. Various segments of the CPU, as shown in Figure 4, will be analized separately allowing the extensive use of block diagrams for improved clarification. KD11D circuit schematics will be referenced throughout the descriptions.

5.2 Data Path

5.2.1 General Description

The simplified KD11D data path consists of five functional units as shown in Figure 3.

Figure 3 DATA PATH



DATA PATH
FIGURE 3

ALU - The heart of the data path is an arithmetic logic unit capable of performing 16 arithmetic or 16 logical (Boolean) operations on the two available 16-bit input words.

AMUX - A four-to-one multiplexer which controls the introduction of new data and the circulation of available data around the data path.

B-Register (BREG) - This 16-bit shift register and its complementary logic is used to store one of the operands required for most ALU operations. It is also needed to implement ROTATE and SHIFT instructions, as well as introducing the constant +1 and generating the sign extended low byte of the B Register contents.

PSW - Eight bit register containing information on the current processor priority, condition codes (N,Z,V and C) describing the results of the last instruction, and an indicator for detecting the execution of an instruction to be trapped during program debugging.

Scratch Pad Memory (SPM) - This 16 word by 16-bit random access memory (RAM) contains eight processor dedicated registers and eight general purpose (user available) registers. Of the eight general purpose registers, one is used as a stack pointer (SP) and another as the program counter (PC).

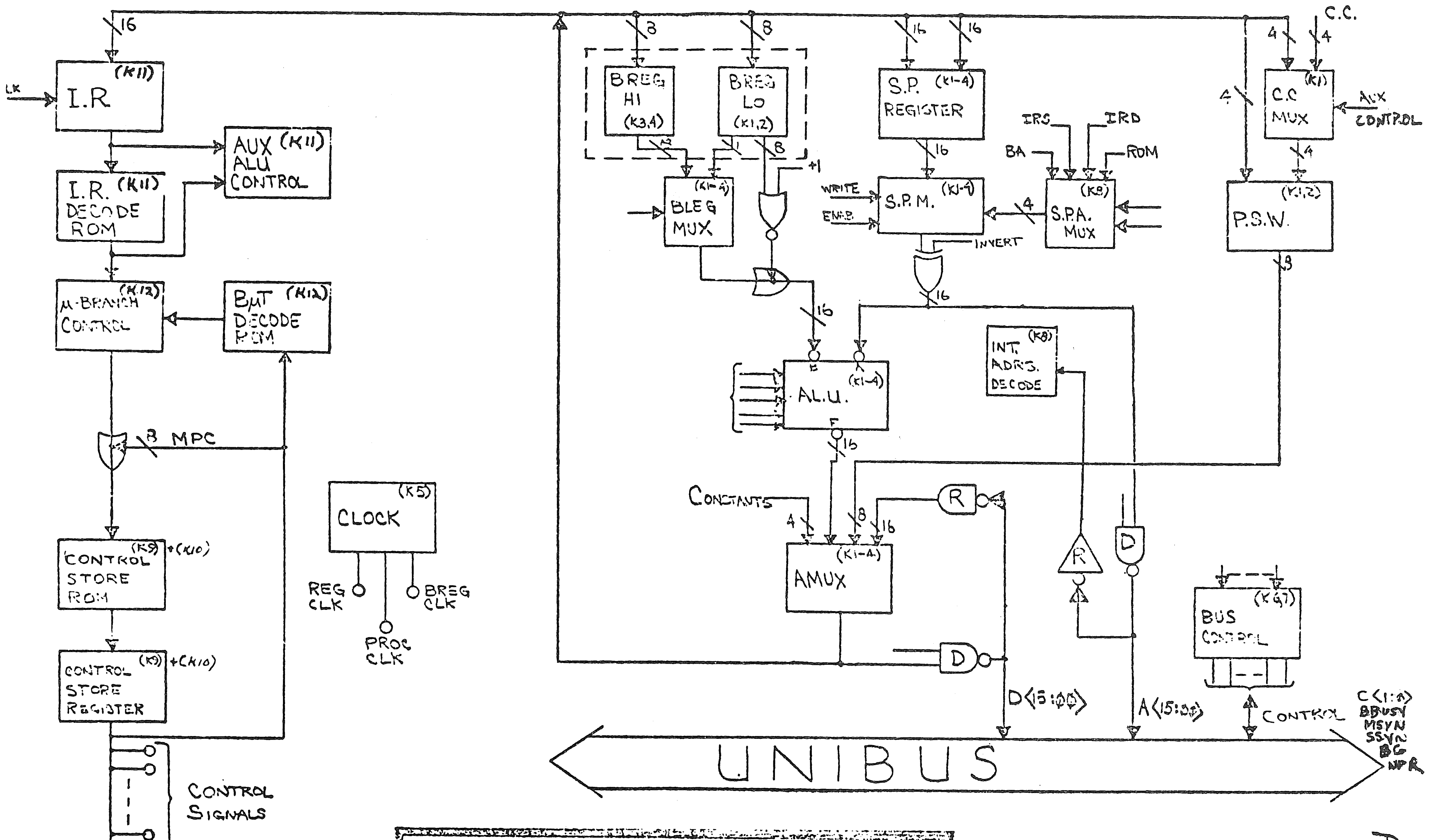


Figure 4 KD11D BLOCK DIAGRAM

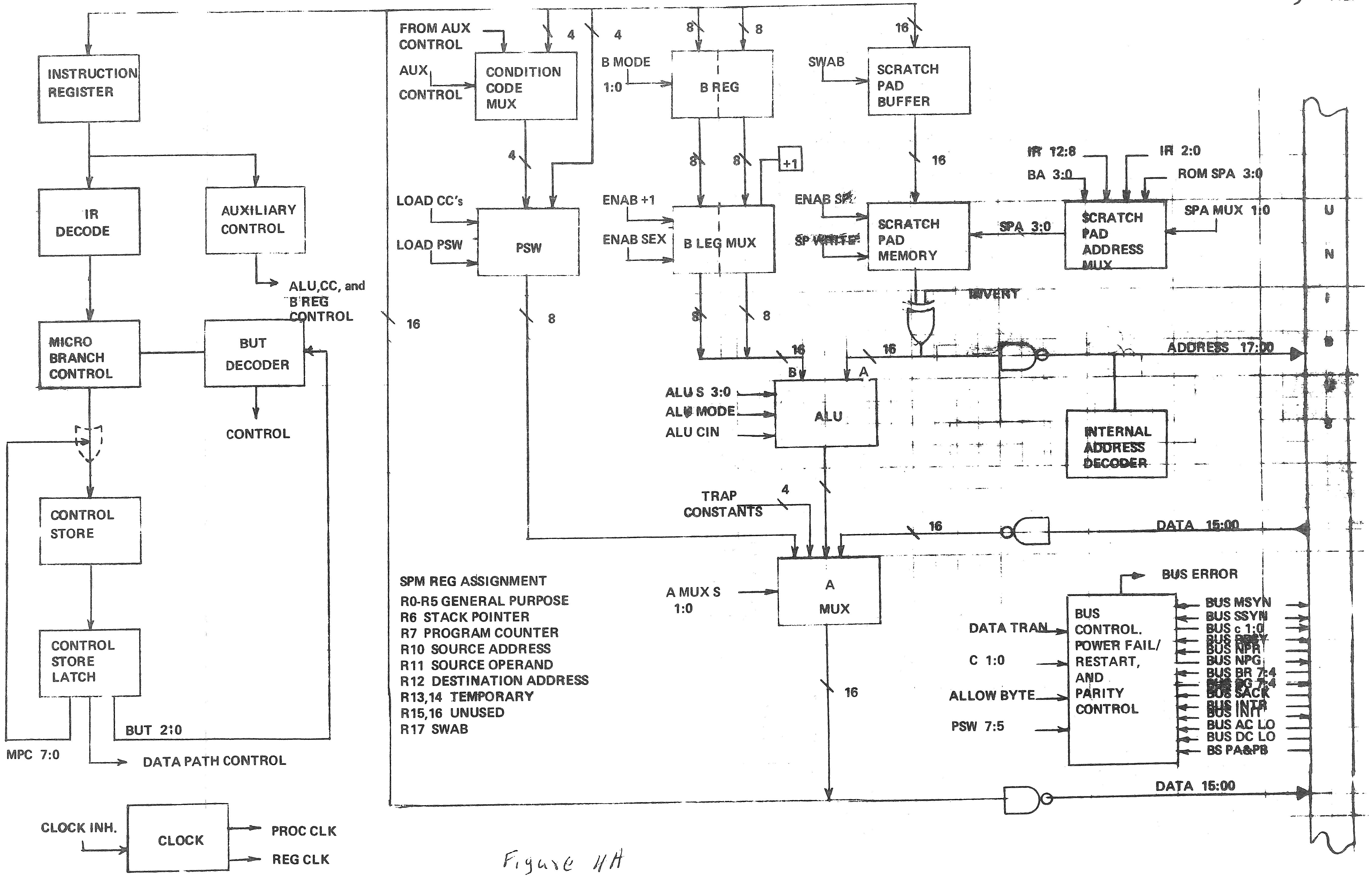

INSTRUCTION REGISTER
IR DECODE
AUXILIARY CONTROL
ALU,CC, and B REG CONTROL
MICRO BRANCH CONTROL
BUT DECODER
CONTROL
CONTROL STORE
CONTROL STORE LATCH
BUT 2:0
MPC 7:0
DATA PATH CONTROL
CLOCK INH.
CLOCK
PROC CLK
REG CLK
FROM AUX CONTROL
AUX CONTROL
CONDITION CODE MUX
4
4
LOAD CC's
LOAD PSW
PSW
8
16
B MODE 1:0
B REG
8
ENAB +1
ENAB SEX
B LEG MUX
+1
SWAB
SCRATCH PAD BUFFER
ENAB SP
SP WRITE
SCRATCH PAD MEMORY
INVERT
IR 12:8
BA 3:0
IR 2:0
ROM SPA 3:0
SPA 3:0
SCRATCH PAD ADDRESS MUX
SPA MUX 1:0
ADDRESS 17:00
INTERNAL ADDRESS DECODER
UNIBUS
B
A
ALU S 3:0
ALU MODE
ALU CIN
ALU
TRAP CONSTANTS
DATA 15:00
A MUX S 1:0
A MUX
BUS ERROR
DATA TRAN
C 1:0
ALLOW BYTE
PSW 7:5
BUS CONTROL. POWER FAIL/ RESTART, AND PARITY CONTROL
BUS MSYN
BUS SSYN
BUS c 1:0
BUS BBSY
BUS NPR
BUS NPG
BUS BR 7:4
BUS BG 7:4
BUS SACK
BUS INTR
BUS INIT
BUS AC LO
BUS DC LO
BS PA&PB
SPM REG ASSIGNMENT
R0-R5 GENERAL PURPOSE
R6 STACK POINTER
R7 PROGRAM COUNTER
R10 SOURCE ADDRESS
R11 SOURCE OPERAND
R12 DESTINATION ADDRESS
R13,14 TEMPORARY
R15,16 UNUSED
R17 SWAB

Figure 4H

Control and Data Flow

Data flow through the Data Path is controlled either directly or indirectly by the CONTROL STORE circuitry (Figure 4) on prints K9 and K10. Each CONTROL STORE Rom location (microinstruction) generates a unique set of outputs capable of controlling the above data path elements and determining the ALU function performed. Sequences of these ROM microinstructions are combined into microroutines which perform the various PDP11 instruction operations. (See section 5.11 for further details).

5.2.2 Arithmetic Logic Unit (Prints K1,K2,K3,K4)

ORGANIZATION

The Arithmetic Logic Unit (ALU) is divided into four 4-bit slices, (K1,K2,K3 and K4 each contain a slice) each consisting of one 4-bit ALU chip (74181) and part of a Look Ahead Carry Generator chip (74182). See Appendix for specifications for the 74181 and 74182 integrated circuits.

INPUTS TO ALU

The A-input to each ALU chip comes from one of the Scratch Pad Memory (SPM) registers as specified by the CONTROL STORE microinstruction being performed (see section 5.2.3 for details). B-inputs to each ALU are specified by the B-Leg Multiplexer logic and can take the forms of either the full 16 bit B Register contents, the lower byte of the B-Register contents sign extended, the constant one (+1) or the constant zero (0) (see section 5.2.4 for further description).

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). The following table lists the ALU functions used in the KD11D and the corresponding bit patterns for the six control signals. Additional functions are shown in Table 12.

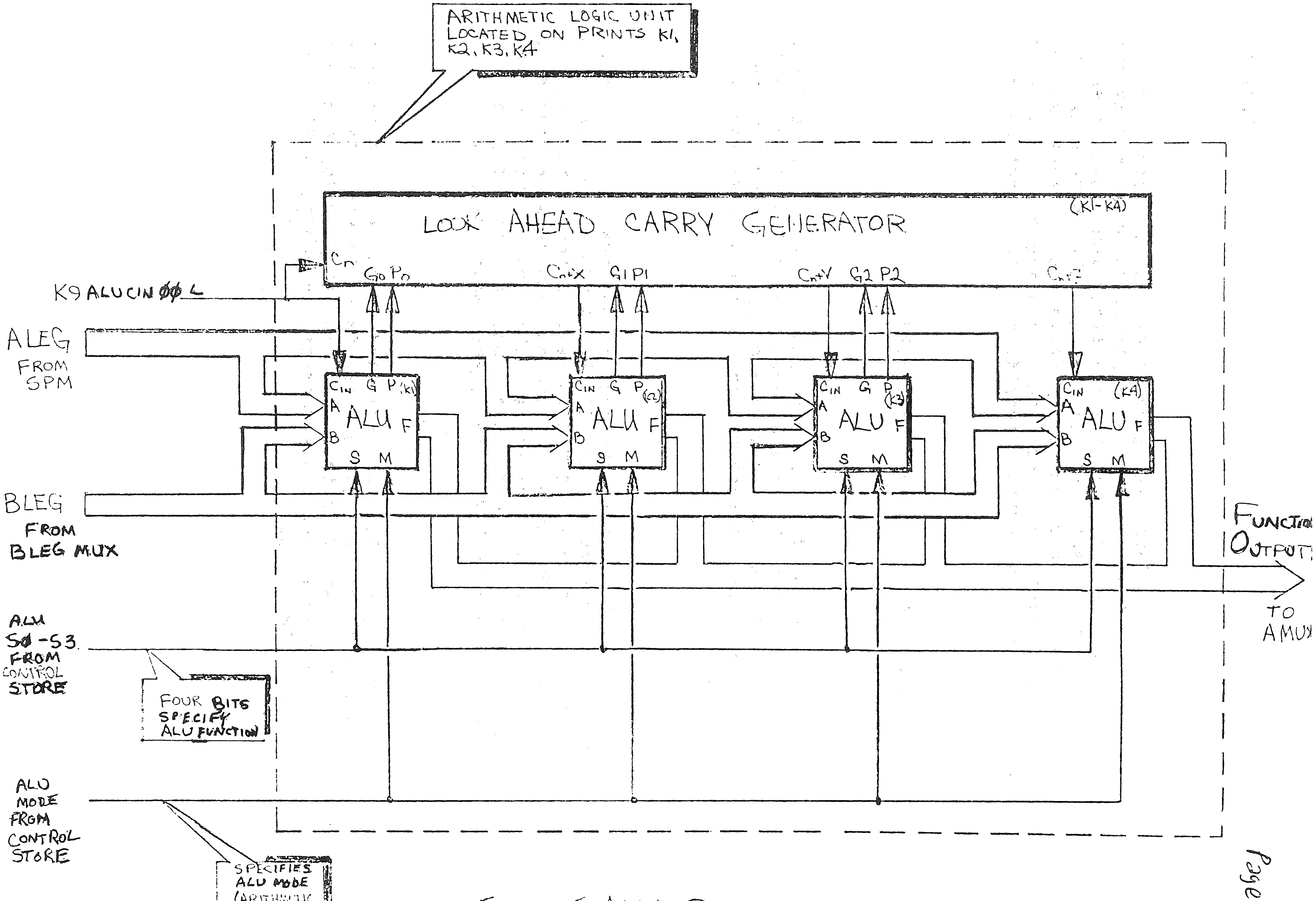

ARITHMETIC LOGIC UNIT LOCATED ON PRINTS K1, K2, K3, K4
LOOK AHEAD CARRY GENERATOR
(K1-K4)
Cn
Go Po
Cn+X
G1 P1
Cn+Y
G2 P2
Cn+Z
K9 ALUCIN ØØ L
ALEG FROM SPM
BLEG FROM BLEG MUX
CIN
G
P
(K1)
(K2)
(K3)
(K4)
A
B
ALU
F
S
M
FUNCTION OUTPUT
TO AMUX
ALU SØ-S3 FROM CONTROL STORE
FOUR BITS SPECIFY ALU FUNCTION
ALU MODE FROM CONTROL STORE
SPECIFIES ALU MODE (ARITHMETIC OR LOGIC)

Figure 5 ALU BLOCK DIAGRAM

ALU FUNCTION	ALU CONTROL SIGNALS					
	M	S3	S2	S1	S0	CIN
0	1	0	0	1	1	1
A	0	0	0	0	0	1
B	1	1	0	1	0	1
A Plus A	0	1	1	0	0	1
A Plus 1	0	0	0	0	0	0
A Plus B	0	1	0	0	1	1
A Plus B Plus 1	0	1	0	0	1	0
A Minus B Minus 1	0	0	1	1	0	1
A + B	0	0	0	0	1	1
A (-B)	0	0	1	1	1	0
A Minus B	0	0	1	1	0	0
A.B Minus 1	0	1	0	1	1	1
-B	1	0	1	0	1	1

Table 10
ALU FUNCTIONS

5.2.3 Scratch Pad Memory

5.2.3.1 Scratch Pad Circuitry

ORGANIZATION

The Scratch pad function is comprised of three functional units (Figure 6) - Scratch Pad Register, Scratch Pad Address Multiplexer and Scratch Pad Memory. The Scratch Pad Register and Scratch Pad Memory are divided into four 4-bit slices one of which is shown on either prints K1, K2, K3, or K4. Scratch Pad Address Multiplexer circuitry is shown on print K8.

DATA INPUT

Data to be written into the Scratch Pad is channeled from the AMUX and clocked into the Scratch Pad Register in either normal form or with high and low bytes reversed (for implementation of the SWAB instruction).

ADDRESSING THE SCRATCH PAD

The Scratch pad Memory (SPM) register address to be accessed is generated by the Scratch Pad Address Multiplexer (SPAM). Depending on the state of the select lines to the SPAM, any of the following can be the source of the address:

a. Bus Address

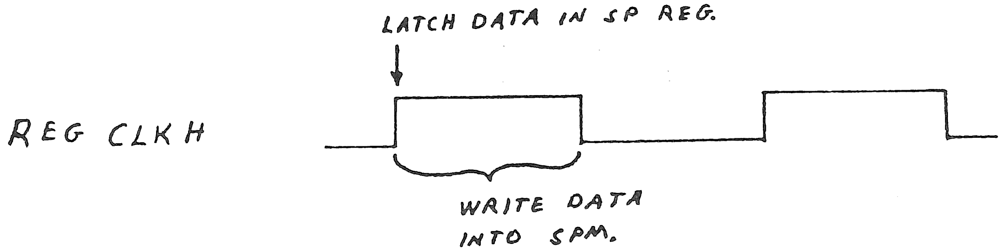
b. Instruction Register Source Field IR11-IR09

c. Instruction Register Destination Field IR05-IR03 or

d. the CONTROL STORE ROM (ROMSPA03-ROMSPA00).

CLOCKING THE SCRATCH PAD

Clocking data from the AMUX lines into the SP Register and writing that data into the SPM, are both accomplished by the REG CLK H clock signal. Data is latched into the SP Register on the rising edge of REG CLK H and is immediately written into the SPM until REG CLK H returns low (logic "0").



SCRATCH PAD CLOCK ENABLES

The K9 SPW HIGH H and K9 SPW LOW H enabling signals generated by the CONTROL STORE determine whether a write operation will be performed during a particular REG CLK cycle. These lines also dictate which bytes (either high or low or both) will be written into the SPM.

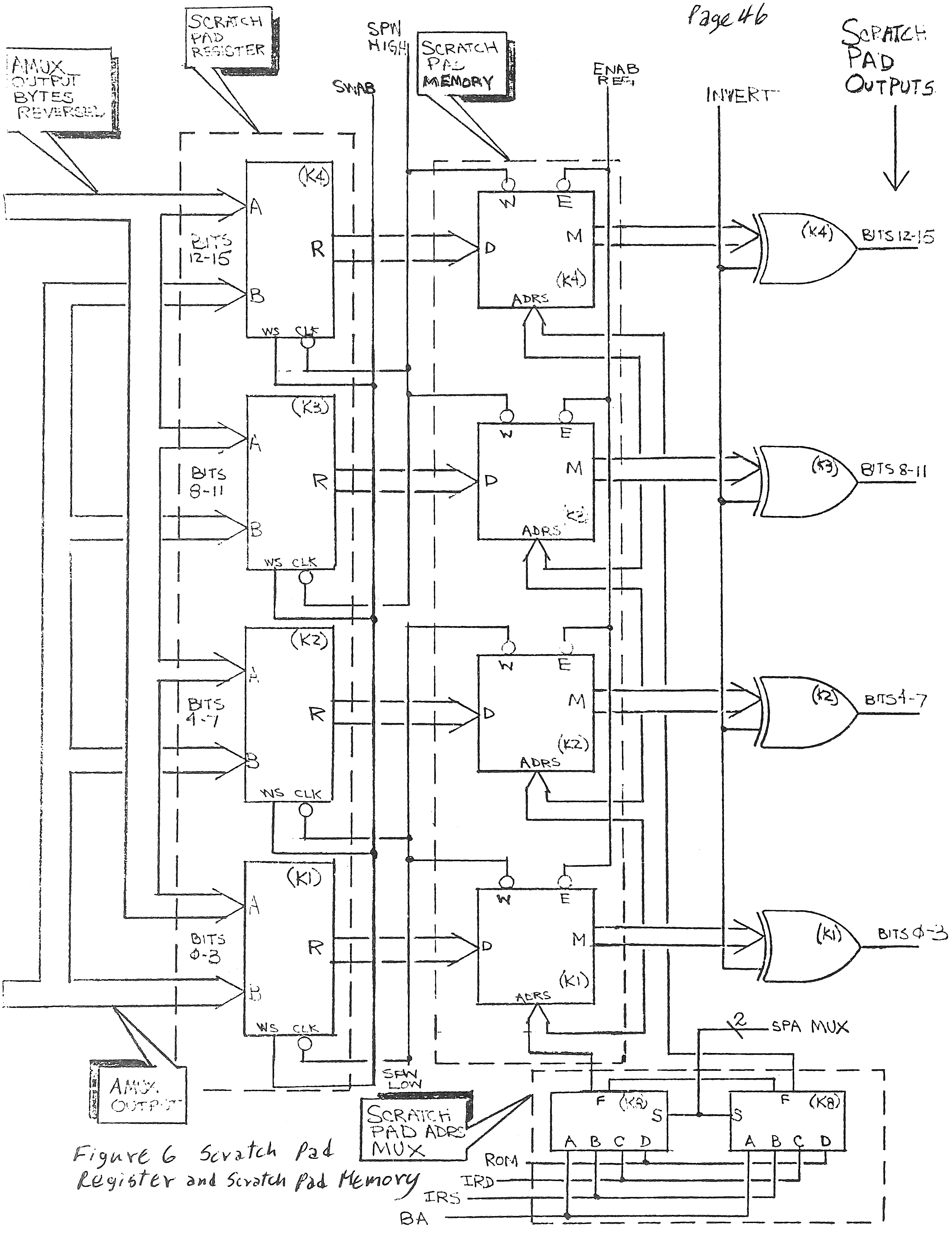


Figure 6 Scratch Pad Register and Scratch Pad Memory

5.2.3.2 Scratch Pad Register (SP REG)

Figure 7 SCRATCH PAD REGISTER

OVERVIEW OF SCRATCH PAD REGISTER

The SP REG consists of four Multiplexer Latch (74298) circuits which allow data from the AMUX lines (AMUXØ5-AMUXØØ) to be latched with the high and low bytes reversed or in normal fashion.

LATCHING OF HIGH AND LOW BYTES

If K1 SWAB L is unasserted (meaning that REGISTER 17 is not being accessed in the SPM), the 74298 B inputs are enabled and data stored in the output of the AMUX. If, however, K1 SWAB L is asserted, the 74298 A-inputs are enabled and the data is stored with the AMUX high and low bytes swapped. This feature is used when performing a SWAB instruction and operating on byte instructions.

The CONTROL STORE outputs K9 SPW HIGH H and K9 SPW LOW H determine whether the low, high or total AMUX lines will be latched into the SP REG at the time K5 REG CLK H occurs. With K9 SPW HIGH H enabled, the high byte of the AMUX will be latched and correspondingly the low byte

Scratch Pad Register

(prints K1, K2, K3, K4)

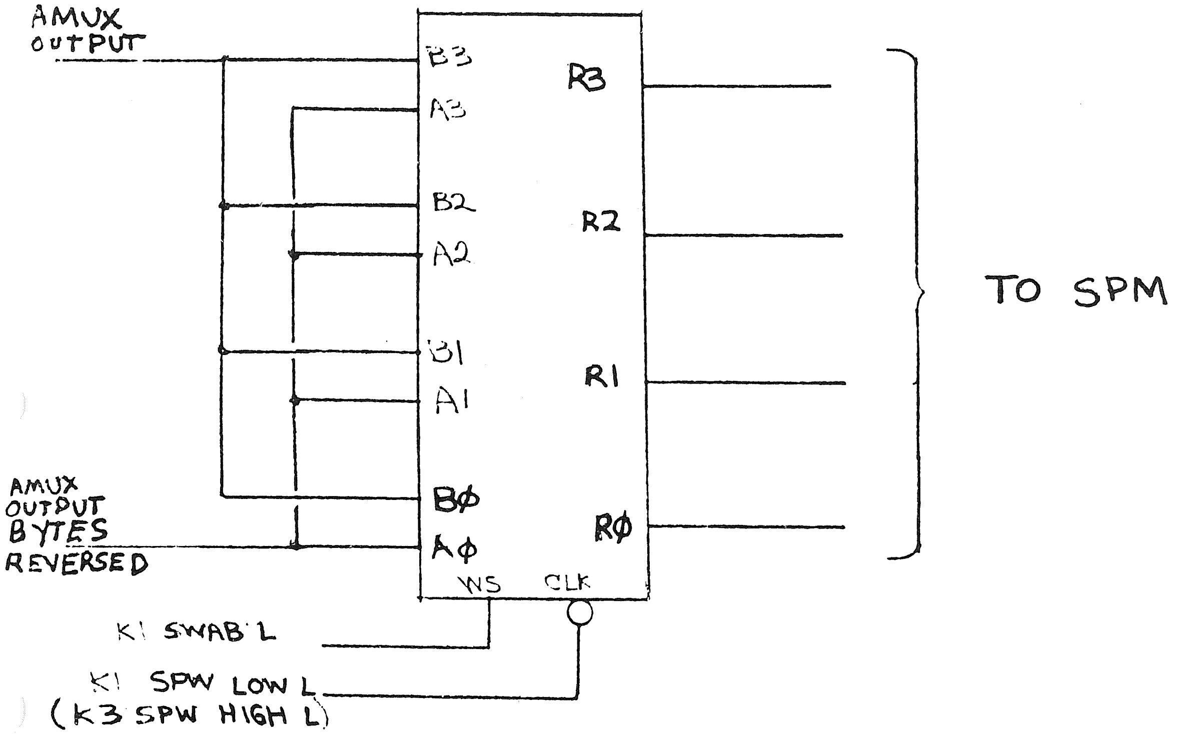


Figure 7 Scratch Pad Register

will be entered when K9 SPW LOW H is true.

5.2.3.2 Scratch Pad Address Multiplexer (SPAM)

SPAM ORGANIZATION

The SPAM generates the four address signals that select the desired SPM word. The SPAM consists of two Type 74S153 Dual 4-Line-to-1 Line Data Multiplexers. The SPAM is shown in print K8. Each of the four 4-line-to-1-line multiplexers (two per 74S153 package) has a common strobe input signal (GND) and common address input signals (KØ SPA MUX ØØ H and KØ SPA MUX Ø1 H). Four data input sources are used and they 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 11 lists the sources of the SPAM input data that are a function of the state of the processor.

Table 11
SPAM Input Data Sources

Function	SPAM Input	Source	Source Print
Source Operand Register Selection	B	Instruction Register Bits Ø6-Ø8	K11
Destination Operand Register Selection	C	Instruction Register Bits ØØ-Ø2	K11
General Purpose Register selection From Console	A	Bus Address Bits ØØ-Ø3	K1
Register Selection By Microprogram	D	Control Store ROM	K1Ø

SPAM SELECT INPUTS

The SPAM address inputs are S1 (signal K1Ø SPA MUX Ø1 H) and SØ (signal K1Ø SPA MUX ØØ H). They are generated by the CONTROL STORE on print K1Ø.

The data input selected is a function of the states of S1 and SØ as shown below.

Address Inputs		
S1	SØ	Output
L	L	A
L	H	B
H	L	C
H	H	D

5.2.3.3 Scratch Pad Memory (SPM)

Figure 8 Scratch pad memory

SPM ORGANIZATION

The Scratch pad Memory (SPM) is a 16-word by 16-bit random access read/write memory composed of four 16-word by 4-bit bipolar (Type 31Ø1A) memory units found on KD11D logic prints K1-K4.

The 16-word by 16-bit organization of this memory provides 16 storage registers that are utilized as shown below.

Page 49A
(prints K1, K2, K3, K4)

SCRATCH PAD MEMORY

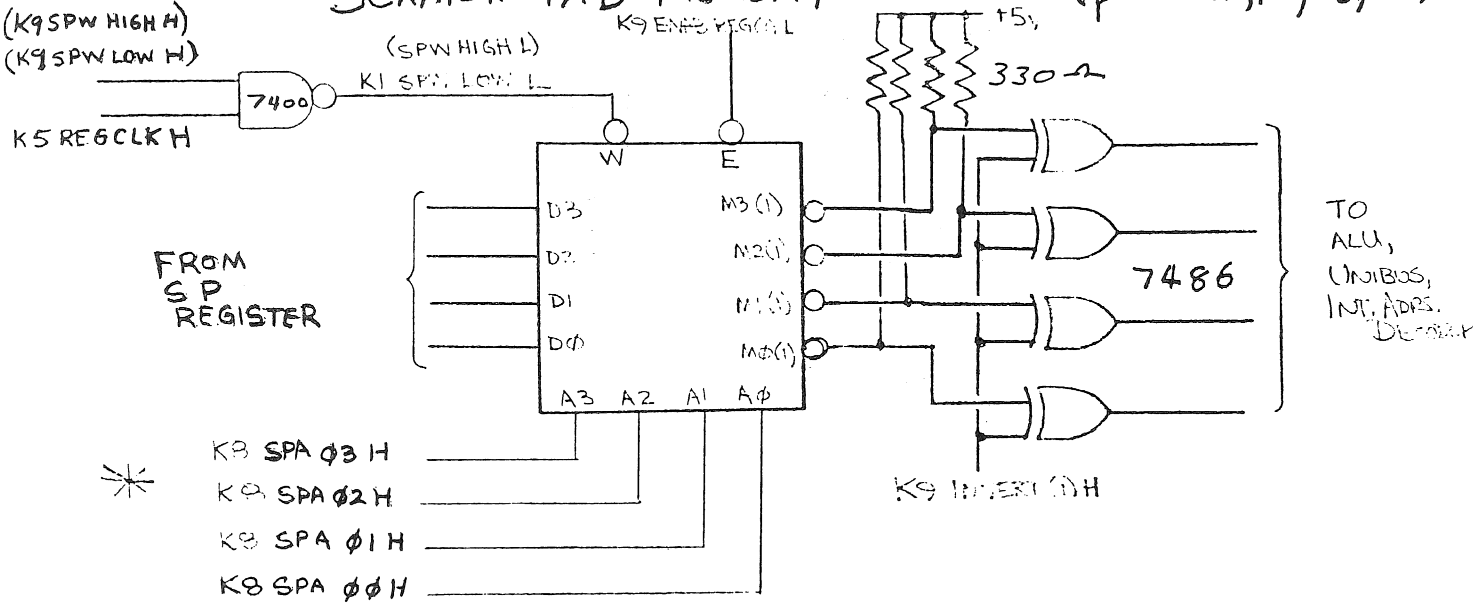


Figure 8 Scratch Pad Memory

FIGURE 9
Register Utilization in SPM

Register Number	Description
RØ	General Purpose Registers
R1	
R2	
R3	
R4	
R5	
R6	Processor Stack Pointer
R7	Program Counter
R1Ø	Source Address Storage
R11	Source Data Storage
R12	Destination Address Storage
R13	Temporary Storage
R14	Unused
R15	
R16	
R17	Temporary Storage for SWAB

SPM DATA OUTPUTS

Data inputs to the SPM are obtained from the previously described SP REG. The output of the scratch pads (31Ø1A) are fed into a set of exclusive - OR (7486) gates which recomplement the memory output data (Data read from the 31Ø1A is always the complement of what was written into it) on read operations.

When the INVERT (1) H signal is used in conjunction with the SPM enable input, ENAB REG (1) L, the exclusive - OR gates allow the AUXILIARY ALU CONTROL circuitry, on print K11 to force (a) all 1's (b) all Ø's (c) the complement of the SPM data or (d) true SPM data onto the ALEG of the ALU.

INVERT (1) H	ENAB REG (1) L	ALU ALEG DATA
Ø	Ø	All 1's
Ø	1	Complement of SPM data
1	Ø	All Ø's
1	1	True SPM data

Figure 1Ø ALU ALEG DATA

Another SPM enable input, WRITE ENABLE, allows either the CONTROL STORE or INTERNAL ADDRESS DECODE circuitry to perform write operations on the SPM. To write into the low byte of the memory, the K9 SPW LOW H signal is enabled and on the next REG CLK H low-to-high transition, the byte is written. Writing into the high byte of the memory is accomplished in a similar manner except that the K9 SPW HIGH H signal is enabled. Full word operations occur when both K9 SPW LOW H and K9 SPW HIGH H are enabled simultaneously.

5.2.4 B Register

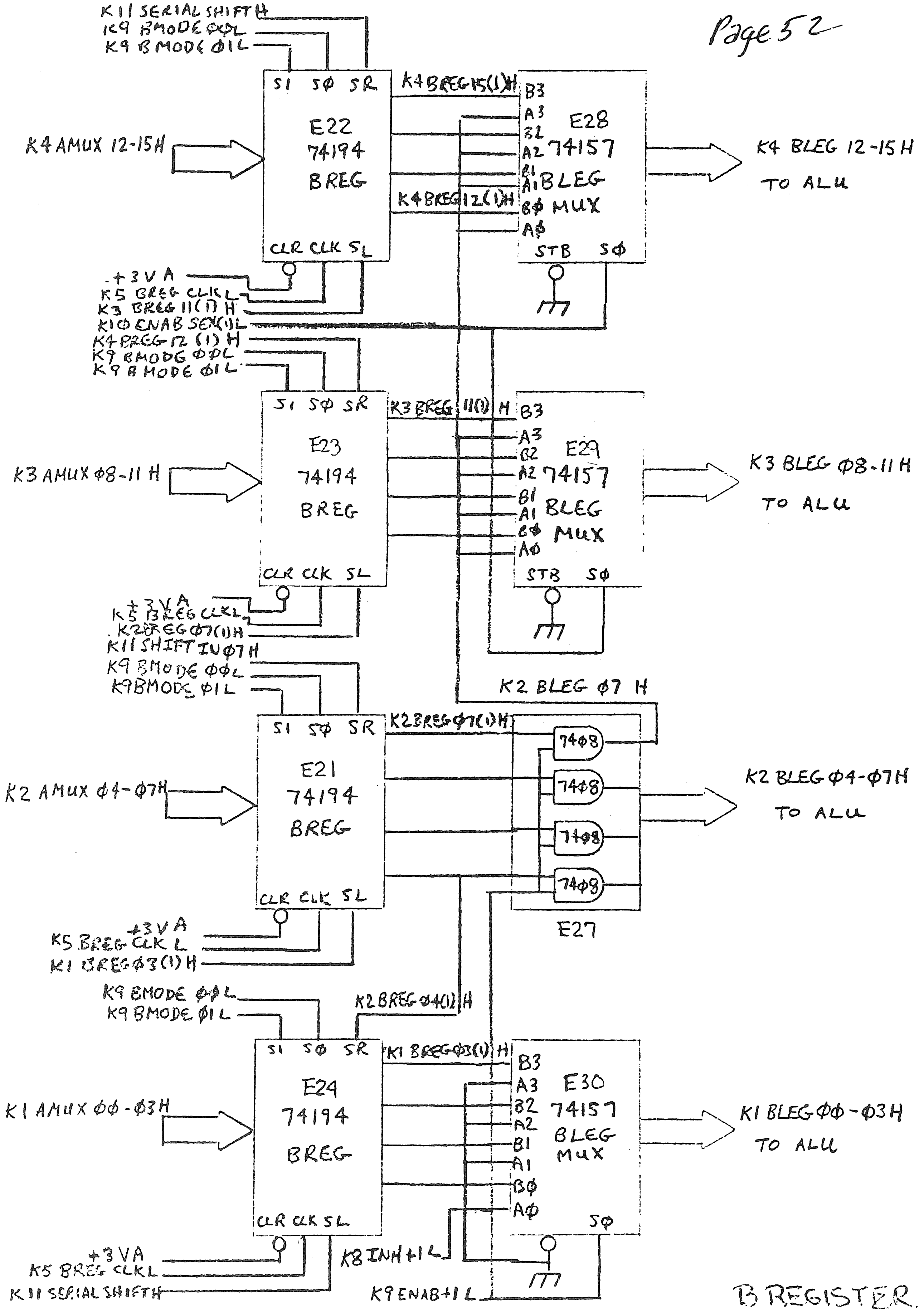
The B Register (B REG) is a general purpose storage register on the B-leg of the ALU consisting of four 4-bit bidirectional shift registers (74194). It is used to store one of the two operands required for most ALU operations and as a shift-left/shift-right register during rotate, shift and byte instructions. Between the BREG and the ALU is a block of multiplexer logic (Figure 11) which performs the following functions.

1. Permits the sign of the operand in the lower byte of the BREG to be extended through the upper byte before it enters the ALU.

2. Can force the constant +1 into the ALU B-leg inputs during operations where a scratch pad register is being incremented or decremented by two.

3. Can force the constant Ø into the ALU B-leg inputs during operations where a scratch pad register is being incremented or decremented by one. (ALU CIN provides for the one)

Data from the AMUX lines, AMUX15-AMUXØØ, can be clocked into the BREG by the K5 BREG CLK L signal.



B REGISTER
FIGURE 11

The type of operation performed by the BREG is determined by the states of mode control inputs 1 and Ø as shown below.

BMode Control		
Ø1	ØØ	Operation
H	H	Parallel Load
L	H	Shift Right (towards LSB)
H	L	Shift Left (towards MSB)
L	L	Hold (clock inhibited)

SOURCE OF BMODE CONTROL

The primary source for the BMODE control signals is the CONTROL STORE (print K9). A wired-OR connection allows these control signals to also be generated by the ROTATE and SHIFT ROM (E87) in the AUX CONTROL logic on print K11.

BREG SHIFT CAPABILITIES

A key to the discussion of the BREG shifting operations is the symbolic representation of the BREG bit structure as shown in Figure 13. Each of the four 74194 Shift Registers which make up the BREG has a shift-left (SL) serial input and a shift-right (SR) serial input which are interconnected in such a way as to create a full 16-bit shift register.

When SL or SR is enabled, the other input is disabled, therefore, depending on the instruction being performed only one serial input will accept the K11 SERIAL SHIFT H signal generated by the ROT/SHFT ROM (E87). Specific SERIAL SHIFT signals are shown in Figure 12.

Instruction	Value of K11 SERIAL SHFT H	Remarks
ASL	GND L	Ø to BREG bit Ø via SL input
ASR	BREG 15 (1) H	Bit 15 of BREG output to bit 15 of BREG via SR input
ROL	COUT (1) H	C bit BREG bit Ø via SL input
ROR	COUT (1) H	C bit to BREG bit 15 via SR input

Figure 12 B REGISTER SHIFT SIGNAL INPUTS

Figure 13 B REGISTER BIT STRUCTURE

BYTE SHIFTS

This register also handles byte shifting as required by instructions ASLB, ASRB, ROLB, and RORB. Signal K11 SHIFT IN Ø7 H is used as a serial right (SR) input to bit Ø7 to handle replication of bit Ø7 for an ASRB instruction and to load the previous contents of the C-bit for an RORB instruction. This signal is also required to perform the word shifting for instructions ASR and ROR because there is no direct connection between bits Ø8 and Ø7 for a shift-right operation. Signal K11 SHIFT IN Ø7 H is generated by BYTE MUX E66 and it represents BREG output bit Ø8 (K3 BREG Ø8 H) during word instructions ASR and ROR.

SPECIFIC SHIFT AND ROTATE OPERATIONS

The shifting requirements for the ASL, ASR, ROL, and ROR instructions are described briefly below.

Arithmetic Shift Left (ASL) - Shifts all bits left one place. Bit Ø loaded with a Ø.

The BREG is shifted left one place. The ROT/SHFT ROM (E87) selects

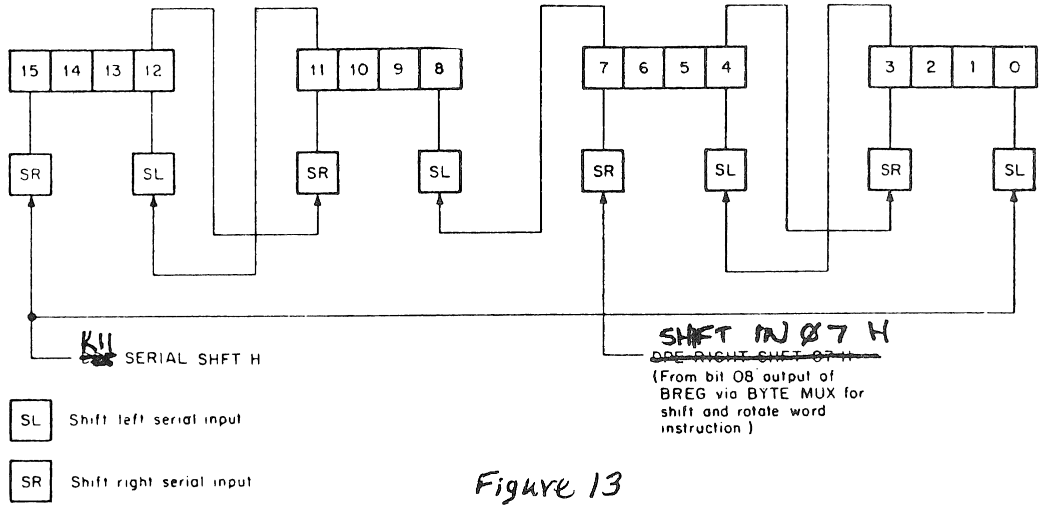

15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
SR
SL
SR
SL
SR
SL
SR
SL
K11 SERIAL SHFT H
SHFT IN Ø7 H
~~DPE RIGHT SHFT 07 H~~
(From bit 08 output of BREG via BYTE MUX for shift and rotate word instruction)
SL Shift left serial input
SR Shift right serial input

Figure 13

B Register Bit Structure

ASL input which is ground. K11 SERIAL SHFT H = Ø and is loaded into BREG bit ØØ via the SL input.

Arithmetic Shift Right (ASR) - Shifts all bits right one place. Bit 15 is loaded with BREG output bit 15.

The BREG is shifted right one place. The ROT/SHFT ROM (E87) selects ASR input [K11 CCNH], which is output bit 15 of the BREG. K11 SERIAL SHFT H equals the bit 15 output of the BREG and is loaded into BREG bit 15 via the SR input. This is replication of bit 15. K11 SHIFT IN Ø7 H from ROT MUX (E66) equals the bit Ø8 output of the BREG and is loaded into BREG bit Ø7 via the SR input to provide the connection from bit Ø8 to bit Ø7.

Rotate Left (ROL) - Rotates all bits left one place. Bit ØØ loaded with C-bit.

The BREG is shifted left one place. The ROT/SHFT ROM (E87) selects ROL input [K1 CBIT (1) H], which is the value of the C-bit prior to execution of the instruction. K11 SERIAL SHFT H equals this value of the C-bit and is loaded into BREG bit ØØ via the SL input.

Rotate Right (ROR) - Rotates all bits right one place. Bit 15 loaded with C-bit.

The BREG is shifted right one place. The ROT/SHFT ROM (E87) selects ROR input [K1 CBIT (1) H], which is the value of the C-bit prior to execution of the instruction. K11 SERIAL SHFT H equals this value of the C-bit and is loaded into bit 15 via the SR input. K11 SHIFT IN Ø7 H equals the bit Ø8 output of the BREG and is loaded into BREG bit Ø7 via the SR input to provide the connection from bit Ø8 to bit Ø7.

In each of these instructions, the C-bit is loaded with a new value from the BREG. This function is discussed in the description of the PSW logic.

BMUX OPERATION

The 16-bit output of the BREG is fed into a set of 2-to-1 multiplexers (Type 74157), and AND gates (Type 74Ø8) as shown in Figure 11. These circuits allow the CONTROL STORE output signals K9 ENAB +1 L and K1Ø ENAB SEX L to control whether the BREG unmodified, BREG sign extended, constant Ø, or constant +1 will be passed on to the ALU B-word inputs. The following truth table shows the various states of these control signals.

K9 ENAB	K1Ø ENAB	ALU BLEG DATA
+1 L	SEX L	
H	H	BREG contents unmodified
H	L	BREG contents sign extended
L	L	Constant +1 or Ø depending on state of K8 IN H +1 L signal.

SIGN EXTENSION OF BREG DATA

When the K9 ENAB +1 L and K10 ENAB SEX L signals request the sign extension of BREG data, the unmodified low byte of the BREG is passed to the ALU along with its highest bit (BREG07) extended (makes ALU BLEG00 thru BLEG15 the same as BLEG07) through the high byte.

CONSTANTS +1 AND 0

The purpose of generating the constants +1 and 0 on the BLEG inputs of the ALU is to aid the processor in performing autoincrement and autodecrement operations. During either operation, if a word instruction is being performed, the specified register is incremented or decremented by two. If however, a byte instruction is being performed, the register is incremented (decremented) only by one. The actual ALU operation is as follows.

RESULT = ALEG DATA + BLEG DATA + ALU CIN

The ALU always uses the K10 ALU CIN 00 L signal to increment or decrement the ALEG input by one; which means that the BLEG input must provide the constant +1 or 0 to obtain the correct autoincrement or autodecrement result for both byte and word instructions.

A BLEG constant +1 is generated by enabling the least significant BLEG bit (BLEG 00) and forcing all other bits (BLEG01-BLEG15) to 0. If a constant 0 is desired, even the least significant bit (BLEG00) is cleared. The actual constant generated is defined by the state of the K8 INH +1 L signal as shown in Figure 11.

The state of the K8 INH +1 L signal is determined by the CONTROL STORE output K10 ALLOW BYTE H and the outputs of the Scratch Pad Address Multiplexer (SPAM) shown in the circuitry on print K8. This logic also prevents the ALU from ever incrementing the PC or SP by one.

5.2.5 AMUX

The AMUX circuitry on prints K1 thru K4 consists of four 4 to 1 Multiplexers (Type 74153) and two 2 to 1 Multiplexers (Type 74157). These circuits can channel either the ALU output data, data received from the UNIBUS, the BUT SERVICE constants (K8 C2 H, K8 C3 H, and K8 C4 H), or the contents of the PSW Register onto the AMUX 00 H thru AMUX 15 H lines which feed the Scratch Pad Register, Instruction Register, B Register and PSW Register. The specific data to be channeled is dependent on the two enable lines K10 AMUX S0 L and K10 AMUX S1 L. Primary source of these control signals is the CONTROL STORE (Print K10). A wire-OR connection capability also allows these signals to be generated by the BUT SERVICE ROM (E71 print K8), and the INTERNAL ADDRESS DECODER ROM (E48 print K8). The following truth table shows the relationship between channeled data and the select lines.

DATA SELECTED	AMUX S0 L	AMUX S1 L
UNIBUS DATA	H	H
BUT SERVICE CONSTANTS	H	L
ALU DATA	L	H
PSW DATA	L	L

5.2.6 Processor Status Word

The processor status word register (PSW) contains information on the current priority of the processor, the result of the previous operation, and indicates a processor trap during debugging. The PSW bit assignments and use are shown in Table 12.

Table 12
Processor Status Word Bit Assignments

Bit	Name	Use
07-05	priority	Set the processor priority.
04	Trace	When set, the processor traps 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 is loaded as a result of instruction execution, program traps, I/O interrupts, and returns to main-line code. In the case of a program trap, interrupt, or return, the PSW is loaded with the second word of the vector from the Unibus data lines via the AMUX. Otherwise, the PSW is loaded through a network of multiplexers and combinational logic that is controlled by the particular instruction being executed.

The PSW is an 8-bit flip-flop register (prints K1 and K2). The condition code bits (N, Z, V, and C) are stored in 74175 quad D-type flip-flop (print K1). The priority bits and T-bit are stored in a 74175 quad D-type flip-flop called PSW 7:4 (Print K2). The output of the T-bit flip-flop is sent to another flip-flop (T DEL) which is used as the trap flag.

The input source for the condition code bits is the output of the condition code multiplexer (CC MUX). The CC MUX (print K1) is a Type 74157 Quad 2-Line-to-1-Line Multiplexer. One of the two 4-bit inputs is selected by the state of the select (S) input. When S is high, the B-input is passed to the D-inputs of the condition code latches (NBIT, ZBIT, VBIT, and CBIT). The B-input consists of AMUX outputs K1 AMUX 00 H-03 H. when S is low, the A-input is selected. The A-input consists of signals from the BYTE MUX (print K10) and the C and V BIT ROM (print K10). These devices are part of the logic used in setting the condition codes as a function of instruction execution and are described in detail in subsequent paragraphs.

The input source for the priority bits (PSW 05-07) consists of AMUX outputs K2 AMUX 05 H-07 H which are sent to D-inputs D1, D2, and D3 of the 74175. Signal K2 AMUX 04 H is sent to D-input D0 of the 74175 as the source of the T-bit.

Each bit of the PSW is clocked by REG CLK H when the CONTROL STORE (print K10) output LOAD PSW L is enabled. The condition code bits N, Z, V, and C can be loaded separately by the same REG CLK H when the CONTROL STORE output LOAD CC L is enabled. The T-bit and PSW<7:4> can also be loaded separately by REG CLK H when the INTERNAL ADDRESS DECODER ROM (E48 print K8) enables EXT LOAD PSW L.

5.3 Condition Codes

The logic necessary for determining the condition codes is shown on print K11 and can be subdivided into three parts as follows.

The condition codes are determined by the CC MUX (print K1) previously discussed, the C and V BIT ROM (print K10), the BYTE MUX (print K10) and the ROT/SHF ROM (print K10). The constraints for each condition code bit are shown in the instruction set specifications of section 3.0.

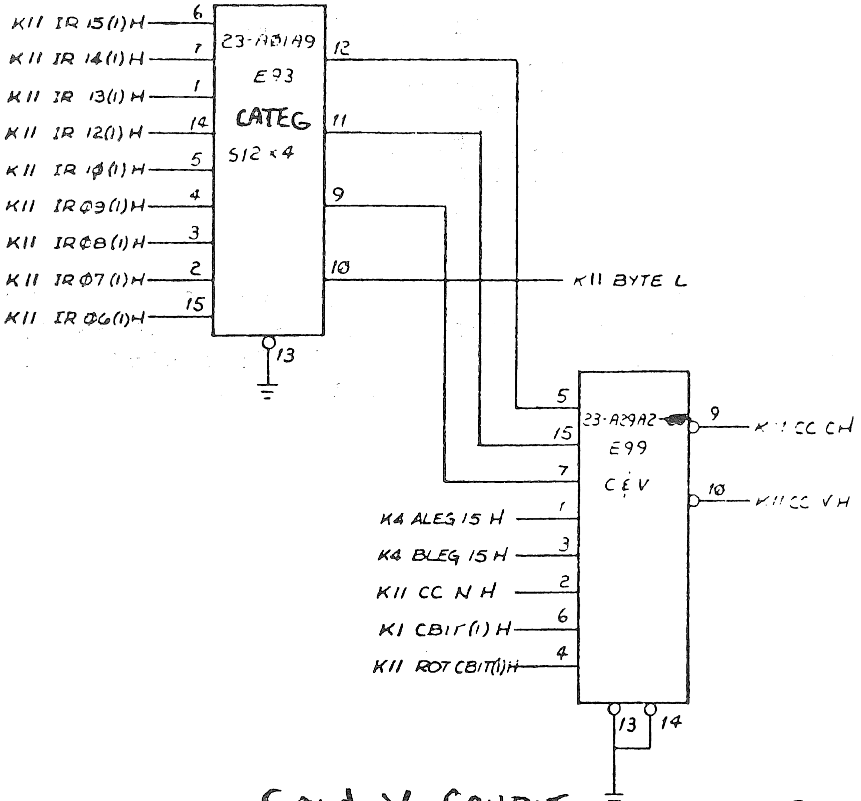
5.3.1 Instruction Catagorizing ROM

The CATEG ROM (E93) on print K11 decodes the instructions in the IR register and catagorizes 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 PDP11 INSTR.
2	INC, DEC
3	CLR,TST,SWAB
4	ADD,ADC
5	NEG,CMP,COM
6	SUB,SBC
7	ROTATES
8	UNUSED

Three of the four outputs of this ROM are used to provide a binary representation of one of the above instruction catagories for the C & V BIT ROM (E99). The fourth output (BYTEL) decodes the fact that the instruction in the IR is a byte instruction.

Figure 14 C AND V CONDITION CODE ROMS



C and V CONDITION CODE ROMS'
FIGURE 14

5.3.2 C & V BIT ROM

The C & V Bit ROM (E99) on print K11 determines the values of the carry and overflow condition code bits as a function of the instruction being performed. Inputs to this ROM come from the ALU ALEG (K4 ALEG 15 H) and BLEG (K4 BLEG 15 H), the ROT SHFT ROM (E87-K11 ROT CBIT (1) H), the PSW (K1 C BIT (1) H), the output of the ALU (K11 CCN H) and the CATEG ROM (E93). Outputs K11 CC C H and K11 CC V H are fed into the CC MUX (E12) on print K1.

5.3.3 Byte Multiplexer

Figure 15 BYTE MULTIPLEXER

The BYTE MUX (E66 print K11) is a 74157 Quad 2 to 1 line multiplexer which determines the N and Z condition code bits and the K11 SHIFT IN Ø7 H signal for the BREG (print K2). A single select input (K11 BYTE L) is used to oppose the A-inputs when a byte operation is performed and the B inputs when not a byte.

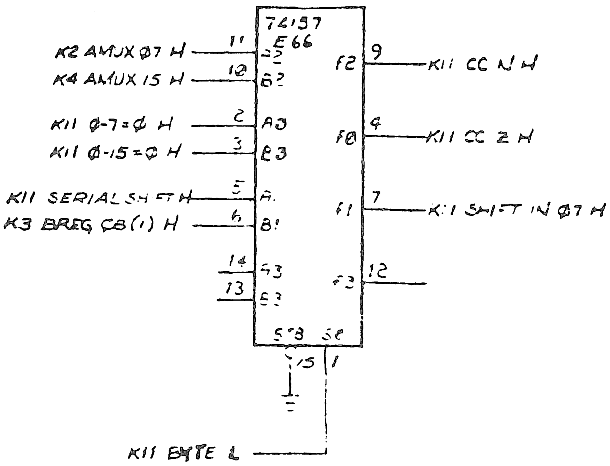
Output K11 CCN H assumes the level of K4 AMUX 15 H when the instruction being performed is a word operation and the level of K2 AMUX Ø7 H when the instruction is a byte. The latter is also possible for instructions performing operations on the high byte of a word because the processor microcode (section 6.Ø) has already swapped the high and low bytes of the input word before the condition codes are detected.

The CC Z H output assumes the level of the K11 Ø-15 = Ø H input when the instruction being performed is a word operation, and K11 Ø-7 = Ø H when the instruction is a byte operation. Both K11 Ø-15 = Ø H and K11 Ø-7 = Ø H are determined by logic on print K11 and the Type 8815 gates connected to the ALU outputs on prints K1, K2, K3 and K4. K11 Ø-7 = Ø H is true if the low byte of the processor operation is zero. K11 Ø-15 = Ø H is true if the 16 bit result is zero.

For shift right operations, the K11 SHIFT IN Ø7 H output assumes the level of the K3 BREG Ø8 (1) H (print K3) input when the instruction performed is a word operation and the level of the K11 SERIAL SHIFT H

Page 60A

BYTE MULTIPLEXER

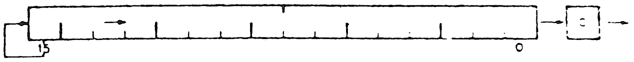


BYTE MULTIPLEXER
FIGURE 15

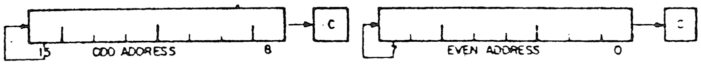
(print K11) output of the ROT/SHFT ROM (E87) for byte operations. To understand the reasons for these signals, the following diagrams indicate the operations performed by the various ROTATE instructions.

ASR
ASRB

Word:

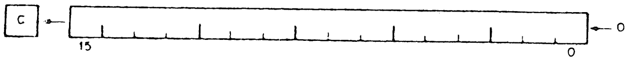


Byte:

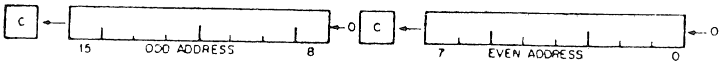


ASL
ASLB

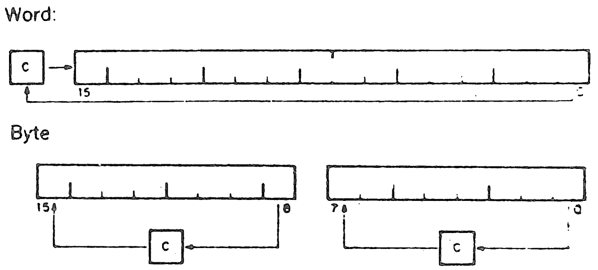
Word:



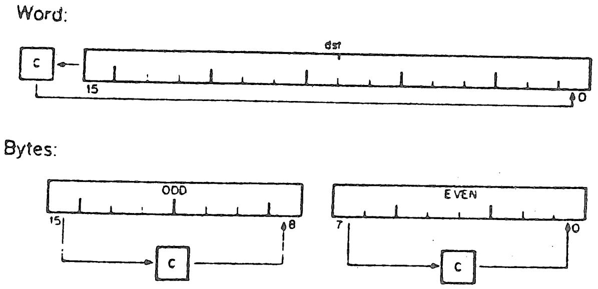
Byte:



ROR
RORB



ROL
ROLB



5.4 UNIBUS ADDRESS and DATA Interface

5.4.1 UNIBUS Drivers and Receivers

Standard bus transceiver circuits Type 8641 are used to interface the processor data path to the UNIBUS address (BUS AØØ-A15) and data (BUS DØØ-D15) lines. These circuits are shown on prints K1 thru K4. A logic diagram for a 8641 is shown below.

Figure 16 UNIBUS TRANSCEIVER

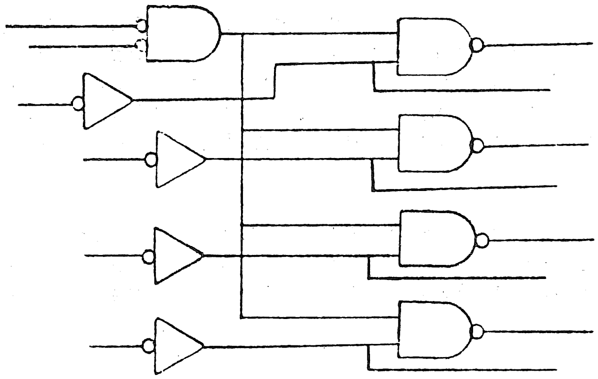
5.4.2 UNIBUS Address Generation Circuitry

A unique feature of the KD11D is that there is no BUS ADDRESS REGISTER. During UNIBUS transfers, bus addresses are obtained directly from the Scratch Pad Memory (SPM) previously discussed. The contents of the selected SPM location is complemented by the Exclusive-OR gates at the outputs of the Scratch Pad and driven onto the UNIBUS by a set of Type 8641 Bus Transceivers (Prints K1, K2, K3 and K4). The driver outputs of these transceivers are enabled by the signal K6 ASSERT ADDRESS L whose source is the data transfer circuitry on print K6.

5.4.3 INTERNAL ADDRESS DECODER

The receiver half of the above mentioned bus transceivers continually monitors the UNIBUS address lines. If the processor is running, these transceivers only allow the INTERNAL ADDRESS DECODER circuit (print K8) to detect transfers to or from the PSW register. While the processor is halted, this decoder circuit enables data transfers between CPU Registers and UNIBUS peripheral devices. A list of these CPU registers and their UNIBUS addresses follows.

PSW	777776	R1Ø	77771Ø
RØ	7777ØØ	R11	777711
R1	7777Ø1	R12	777712
R2	7777Ø2	R13	777713
R3	7777Ø3	R14	777714



UNIBUS TRANCEIVER
FIGURE 16

R4	777704	R15	777715
R5	777705	R16	777716
R6	777706	R17	777717
R7	777707		

One point of clarification that should be noted, is that while the CPU is running, only the PSW can be accessed through its UNIBUS address; the General Registers cannot be accessed in this manner. While the processor is halted all CPU Registers and the PSW can be accessed through UNIBUS addressing.

5.4.5 UNIBUS DATA Transceivers

The Data Path circuitry also contains UNIBUS Transceivers Type 8641 (Prints K1, K2, K3 and K4) which send and receiver data from the UNIBUS data lines D00-D15. The receiver section of these circuits inputs data to the D-inputs of the AMUX (Figure 4) where it may be channeled to either the Instruction Register (IR), B Register, or PSW upon request.

The driver sections of these transceivers obtains data from the AMUX output lines AMUX00-AMUX15 (prints K1 thru K4) and drives it onto the UNIBUS when the signal ENAB DATA L is generated by the DAT TRAN circuitry (print K6).

5.5 Instruction Decoding

5.5.1 General Description

Two methods are used to control instruction decoding. One uses microroutine selection and the other uses 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_RY 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 (Section 5.11) and the other is a knowledge of the PDP-11 instruction format (Section 3.0).

Certain facts concerning the PDP-11 instruction set are listed below.

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

2. There are a number of instructions that require two address calculations and a larger number that 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-D 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 program status word.

5.5.2 Instruction Register

Each PDP-11 instruction obtained from memory is stored in the 16-bit INSTRUCTION REGISTER (IR) on print K11. This register consists of three 6-bit D-Type Registers (Type 74174) and one D-Type Flip-Flop (Type 7474). The purpose of the IR is to store the instruction for the complete instruction cycle so the IR DECODE (print K12) and AUXILIARY ALU CONTROL (print K11) circuits can decode the correct control signals throughout the instruction cycle.

The IR latches data from the AMUX00-AMUX15 (prints K1 thru K4) lines on either the trailing edge of K5 SERV IR H or on K10 LOAD IR L and the trailing edge of K5 BREG CLK L.

When K5 PROC INIT H occurs, all the IR bits except K11 IR 15(1) H are cleared; K11 IR 15(1) H is set by K5 PROC INIT L. This means that the IR DECODER circuit will interpret this new IR output as a conditional branch while K5 PROC INIT H is true. This prevents processor from decoding a HLT instruction on any INITIALIZE condition.

If a trap instruction is loaded into the IR and decoded, it is necessary to clear that instruction from the IR before the micro-PC goes to the next SERVICE routine. Failure to do this will cause the SERVICE routine to loop on the trap instruction. The BUT SERVICE PROM (E71 print K8) asserts K8 INST TRAP SER L which in turn causes K5 SERV IR H. On the trailing edge of K5 SER IR H, K11 IR15 (1) H is set by K8 INST TRAP SER L and all other bits of the IR are loaded with zeros from the AMUX lines. This results in a conditional branch instruction being loaded into the IR.

If a BUS ERROR (BE) occurs while the CONTROL STORE output signal ENAB DBE L is asserted, the whole IR register is cleared (PDP-11 Halt) causing the processor to automatically halt. Bus errors occuring without the ENAB DBE L signal have no effect on the IR.

5.5.3 Instruction Decoder

5.5.3.1 Instruction Decoder Circuitry

The INSTRUCTION DECODE and CONTROL STORE ROM circuitry on print K9, K1Ø, K11 and K12 could be thought of as an internal microprocessor which interprets PDP-11 instructions and translates them into a set of microinstructions each consisting of 38 control signals. These control signals then determine the operation of the data path and UNIBUS control circuitry.

A block diagram of the CONTROL STORE and INSTRUCTION DECODFR is shown in Figure 4. Note that all outputs of the CONTROL STORE ROMS (prints K9 and K1Ø) are latched in Hex D Type Registers (Type 74174).

Eight of these latched signals (K9 MPC Ø7 L-K9 MPC ØØ L) are fed back to the inputs of the CONTROL STORE ROMS as the next micro-instruction address and can thus be called the micro-PC. The wire-OR capability of these lines allows the IR DECODER circuitry to force microbranching addresses on certain enabling conditions. The actual microbranch address will be dependent on the instruction being decoded, the instruction mode used (MODES Ø-7), and the operand required (source or destination).

The INSTRUCTION DECODER circuitry is shown on print K12. It consists of seven 256x4 bit ROMs and several Type 74HØ1, Type 74Ø2 and Type 74Ø4 logic gates. To better understand the operation of this logic, the following descriptions are based on instruction types.

5.5.3.2 Double Operand Instructions

Double operand instructions require two address calculations, one for the source and one for the destination operand. The micro-branch to the sequence of microinstructions which determine the source operand is initiated by the CONTROL STORE output signal K9 IR DECODE (1) H. When this signal is enabled, the IR DFCODER Rom DOP DECODE (E69) (Print K12) checks the instruction in the IR (OP CODE bits IR14-12). If the instruction is a double operand type, the ROM outputs are asserted as follows:

	ROM OUTPUTS			
TYPE	K12	K9	K9	K9
INSTRUCTION	IR CODE ØØ L	MPC Ø5 L	MPC Ø4 L	MPC Ø3 L
Double Operand Inst.	1	Ø	Ø	1
Reserved Inst. (EIS)	Ø	1	1	1
Other Instructions	1	1	1	1

Coupled with the micro-PC outputs of the DOP DEC ROM are the outputs of a set of Type 74HØ1 gates on print K12. These gates when enabled place the contents of the source mode field (IR11-IRØ9) of the PDP11 instruction being decoded on the MPC ØØ L-MPC Ø2 L lines. These gates are enabled only when the instruction being decoded is of the double operand type (K12 IR12-14=0 H true), the K9 IR DECODE (1) H signal is asserted and the instruction is not reserved (K12 IR CODE 00 L unasserted).

A summary of the various source micro addresses is shown below.

INSTRUCTION	SOURCE MODE	OCTAL MICRO BRANCH ADDRESS
DOP	0	60
	1	61
	2	62
	3	63
	4	64
	5	65
	6	66
	7	67
RESERVED DOP		00

Note that a ground on the MPC lines represents a logic "1" (negative logic).

The DOP DEC ROM described above is also used to decode the micro-PC address for the various CONTROL STORE destination operand routines. When the K9 BUT DEST L input is asserted by the Control Store circuitry, the DOP DEC ROM decodes the instruction, determines if it is a modifying or non-modify instruction and asserts either the address 005(8) or 006(8) on the K9 MPC 05-K9 MPC 03 lines. If a MOV instruction is decoded and the K12 DM0 H (destination Mode 0) input is asserted, the micro address 001(8) is placed on the K9 MPC 03-K9 MPC 05 lines.

Similar to the circuitry described above for micro-addressing the source operand routine, a set of Type 74H01 gates on print K12 are also used to decode the destination mode field (K11 IR 03 (1) H - K11 IR 05 (1) H) of the instruction being decoded and place its contents on the K9 MPC 00 - K9 MPC 02 lines when enabled. For double operand instructions, enabling occurs when the CONTROL STORE asserts K12 BUT DEST L.

A summary of the various destination micro-addresses is as follows.

INSTRUCTION	DESTINATION MODE	OCTAL MICRO-BRANCH ADDRESS
MODIFY INSTRUCTIONS (ADD,SUB,BIC,BIS, and MOV not DM0)	0	40
	1	41
	2	42
	3	43
	4	44
	5	45

	6	46
	7	47
NON MODIFY INSTRUCTIONS (CMP,BIT)	Ø	5Ø
	1	51
	2	52
	3	53
	4	54
	5	55
	6	56
	7	57
MOV DESTINATION MODE Ø INSTRUCTIONS	Ø	1Ø

5.5.3.3 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 256x4 Bit ROMs, SOP MICRO BRANCH (E81) and SOP DEC (E75), both on print K12.

The SOP MICRO BRANCH ROM (E81) monitors the necessary IR input lines and asserts the correct micro-PC address on lines K9 MPC Ø3 - K9 MPC Ø5 when the K9 IR DECODE L signal is asserted and the SOP enable signal K12 IR 12-14=Ø L is true. The K12 DEST L output is also activated when a 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.

The SOP MICRO BRANCH ROM is also used to decode JSR instructions. This decoding is performed exactly as described above for SOP instructions. The K12 DMØ H input to the ROM is used to detect the illegal instruction JSR destination mode Ø. When this occurs, no micro-pc address is allowed on the ROM outputs.

INSTRUCTION	DESTINATION MODE	MICRO BRANCH ADDRESS
SOP MODIFY INSTRUCTIONS (CLR,COM,INC,DEC,NEG,ROTATE AND SHIFT INST.)	Ø	4Ø
	1	41
	2	42
	3	43
	4	44
	5	45
	6	46
	7	47
SOP NON MODIFY INSTRUCTIONS (TST)	Ø	5Ø
	1	51
	2	52

	3	53
	4	54
	5	55
	6	56
	7	57
JSR INSTRUCTIONS	Ø	ØØ
	1	21
	2	22
	3	23
	4	24
	5	25
	6	26
	7	27

THE SOP DEC ROM monitors the same input signals as the SOP BRANCH ROM. Its purpose however, is to decode illegal, reserved and trap instructions. The three output signals IR CODE ØØ L - Ø2 L are enabled as follows.

	IR CODE		
INSTRUCTIONS	Ø2	Ø1	ØØ
RESERVED INSTRUCTIONS	1	1	Ø
ILLEGAL INSTRUCTION (JSR MODEØ)	1	Ø	1
EMT INSTRUCTIONS	Ø	1	Ø
TRAP INSTRUCTIONS	Ø	Ø	1

5.5.3.4 Branch Instructions

Conditional branch instructions are completely decoded by the BRANCH DEC ROM (E6Ø) on print K12. This Rom is enabled when IR bits IR11-IR14 are all low (IR11-14=Ø L) and the IR DECODE L signal is active. The input lines monitored are the four condition code bits (N,Z,V and C) and four IR bits (IR15,1Ø,9,8). When a branch is decoded, the MPC Ø6 L output signal is enabled. The branch instruction microcode routine in the CONTROL STORE will sign extend the branch off-set and shift it left one place.

5.5.3.5 Operate Instructions

There are three 256x4 Bit ROMs in the instruction decoding circuitry for decoding PDP11 operate instructions. These ROMs are T BIT DEC, TRAP DEC, and OP BRANCH which are found on print K12.

The OP BRANCH ROM (E82) monitors the IR output lines IRØØ (1) H - IRØ7 (1) H. It is enabled when IRØ8 (1) H thru IR15 (1) H are all low (IRØ8-15=Ø L) and IR DECODE L is active. The PDP11 operate instructions are decoded into the following micro-pc addresses on the ROM outputs MPC 00 L - MPC 02 L.

INSTRUCTION	MICRO BRANCH ADDRESS
RESET	2
RTI	3
SET CONDITION CODES	4
CLEAR CONDITION CODES	5
RTS	6
WAIT	7

The T BIT DEC ROM (E76) has the same inputs and enables as the OP BRANCH ROM. Its purpose is to decode RESET, RTT, and RTI instructions and activate the outputs START RESET L and ENAB TBIT L accordingly.

The TRAP DEC ROM (E70) again has the same inputs as the previous two ROMs. Its purpose is to decode HALT, reserved, trap and illegal instructions and enable the outputs accordingly.

	IR CODE		
INSTRUCTION	02	01	00
RESERVED INSTRUCTIONS	1	1	0
ILLEGAL INSTRUCTIONS	1	0	1
BPT INSTRUCTIONS	1	0	0
IOT INSTRUCTIONS	0	1	1
HALT INSTRUCTIONS	Enable HLT RQET L		

5.6 Auxiliary ALU Control

The AUX Control circuitry on the KD11D consists of three bipolar ROMs shown on print K11.

ROM	NAME	
32X8 Bit	AUX DOP	E94
256X8 Bit	AUX SOP	E89
256X4 Bit	ROT/SHFT	E87

These ROMs determine the ALU operation to be performed whenever the microcode executes the action X_Y OP B where Y designates a scratch pad register and X designates either Register B or a scratch pad register.

The AUX DOP ROM decodes double operand instructions and is enabled by the CONTROL STORE signal 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 and A represents any scratch pad register.

			ROM OUTPUTS					
INSTRUCTION	OPERATION	INVERT	S3	S2	S1	SØ	CIN	MODE
MOV (B)	B_A	1	Ø	Ø	Ø	Ø	1	1
CMP (B)	B_A MINUS B	1	1	Ø	Ø	1	Ø	Ø
ADD	B_A PLUS B	1	Ø	1	1	Ø	1	Ø
SUB	B_-A PLUS B PLUS 1	Ø	Ø	1	1	Ø	Ø	Ø
BIT (B)	A.B	1	Ø	1	Ø	Ø	1	1
BIC (B)	B_(-A).B	1	1	1	Ø	1	1	1
BIS (B)	B_A+B	1	Ø	Ø	Ø	1	1	1

The AUX SOP ROM decodes single operand instructions and is enabled by the CONTROL STORE signal AUX SETUP H. The following table expresses the ROM outputs as a function of the SOP instructions decoded.

		ENAB				ROM OUPUTS			
INSTRUCTION	FUNCTION	REG	INVERT	S3	S2	S1	SØ	CIN	MODE
CLR (B)	B_Ø	Ø	1	1	1	Ø	Ø	Ø	Ø
COM (B)	B_-B	Ø	1	1	Ø	1	Ø	Ø	1
INC (B)	B_Ø PLUS B PLUS 1	Ø	1	Ø	1	1	Ø	Ø	Ø
DEC (B)	B_(1.B) MINUS 1	Ø	Ø	Ø	1	Ø	Ø	1	Ø
NEG (B)	B_Ø MINUS B	Ø	1	1	Ø	Ø	1	Ø	Ø
TST (B)	B_B	Ø	1	Ø	1	Ø	1	1	1
ADC (B)	B_Ø PLUS B PLUS CIN	Ø	1	Ø	1	1	Ø	1	Ø
SBC (B)	B_(1.B) MINUS 1 PLUS -C	1	Ø	Ø	1	Ø	Ø	Ø	Ø

The INVERT H and ENAB REG L outputs are used to create the Ø and 1 inputs on the ALEG of the ALU as described previously in the ALU section.

Auxiliary control signals are also necessary for performing rotate and shift operations. The ROT/SHFT ROM on print K1Ø decodes these instructions and outputs those control signals required to shift the contents of the BREG. Inputs BREG ØØ(1) H, CC N H, and CBIT (1) H also determine the SERIAL SHIFT H and ROT CBIT (1) H signals. The SERIAL SHIFT H signals is sent to the BYTE MUX (print K1Ø) where it is used in determining the SHFT IN 07 H signal used in the B REG shifting operation. ROT CBIT (1) H is used in the calculation of the new carry condition (C & V BIT ROM). 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 setup without slowing the processor.

A summary of the AUXILIARY CONTROL is shown in the Table enclosed.

TABLE 12

Auxiliary Control for Binary and Unary Instructions

Inst.	Condition Codes			ALU Function	CIN	B
	N and Z	V	C			
MOV (B)	Load	Cleared	Not Effected	A Logical	0	Load
CMP (B)	Load	Load like SUBTRACT	Load like SUBTRACT	A - B	[illegible]	Load
BIT (B)	Load	Cleared	Not Effected	A • B Logical	0	Load
BIC (B)	Load	Cleared	Not Effected	~A • B Logical	0	Load
BIS (B)	Load	Cleared	Not Effected	A+B Logical	0	Load
ADD	Load	Set if OP's same sign and result different.	Set if carry out	A plus B	0	Load
SUB	Load	+ - (-) = - - (-) (+) = + } Set	Set if Carry	A plus B	+1	Load
CLR (B)	Load	Cleared (like ADD)	Clear	0	0	Load
COM (B)	Load	Cleared	Set	~B Logical	0	Load
INC (B)	Load	Set if dst held 100000 before OP	Not Effected	A plus B	+1	Load
NEG (B)	Load	Set if result is 100000	Cleared if result is 0; set otherwise	A - B	[illegible]	Load
ADC (B)	Load	Set if dst was 077777 and C = 1.	Set if dst was 177777 and C = 1.	A plus B ~~A Arithmetic~~	+C	Load
SBC (B)	Load	Set if dst was 100000.	~~Cleared~~ Set if dst was 0 and C = 1; ~~set~~ Cleared otherwise.	(A·B) MINUS1	~C	
TST (B)	Load	Cleared	Cleared	B Logical	0	Load
ROR (B)	Z ← (C:01) N ← C	N ⊕ C	(0)			Shift Right
ROL (B)	Z (14:C) N ← (14)	N ⊕ C	(15) B (7)			Shift Left
ASR (B)	Z ← (15:01) N ← N	N ⊕ C	C ← (15)			Shift Right
ASL (B)	Z ← (14:01) N ← (14)		C ← (15)			Shift Left

5.7 Data Transfer Circuitry

5.7.1 General Description

All UNIBUS data transfers are controlled by the DAT TRAN circuitry on print K6. This logic monitors the busy status of the UNIBUS, controls the processor bus control lines BBSY, MSYN, C1 and CØ, and detects PARITY ERRORS (PE), BUS ERRORS (BE) and EOT ERRORS (EOT).

5.7.2 Control Circuitry

5.7.2.1 Processor Clock Inhibit

All processor data transfers on the UNIBUS are initiated by the CONTROL STORE output K1Ø DAT TRAN (1) H (print K1Ø). This signal combines with the signal K6 EOT (Ø) H (normally a logic "1" between transfers) to create K6 TRANS INH L stopping the processor clock.

5.7.2.2 UNIBUS Synchronization

The synchronizer logic shown in Figure 17 (from print K6) arbitrates whether the processor or some other UNIBUS peripheral will control the UNIBUS.

FIGURE 17 DATA TRANSFER SYNCHRONIZER

A logic "1" level (+3v) on the set input of the E121 flip-flop specifies that the bus is presently in use. Each of the inputs which combine to create this level monitors a specific set of bus conditions.

NPR - A UNIBUS peripheral has aserted a Non Processor Request (NPR) and wishes to gain control of the bus

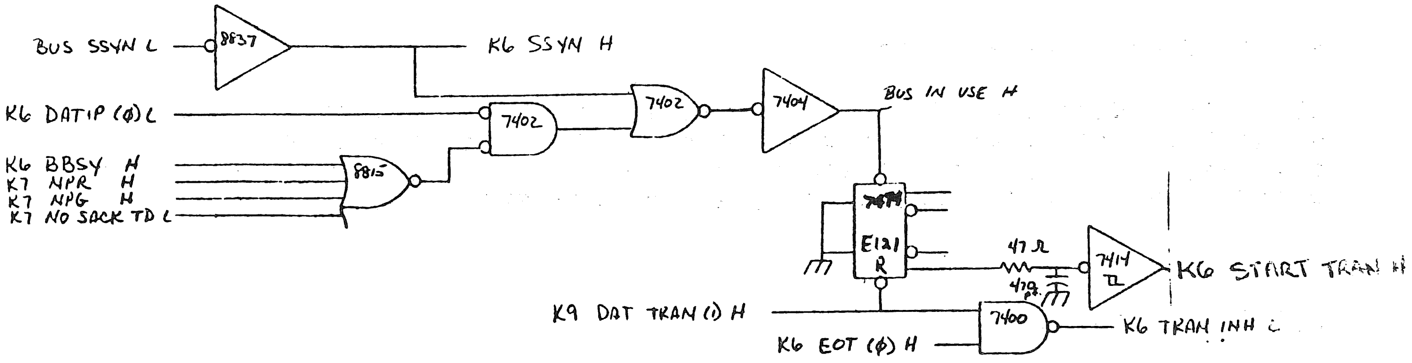


Figure 17

DATA TRANSFER SYNCHRONIZER

immediately.

BBSY - Another UNIBUS peripheral already has control of the bus and is asserting a bus busy (BBSY) signal.

NPG - An NPR device has requested control of the UNIBUS and the KD11D processor has issued a non-processor request grant (NPG). The condition may exist where the NPR device has already recognized the NPG and has dropped its NPR signal while not having asserted a SACK or BBSY yet.

NO SACK TD L - An NPR device has requested control of the UNIBUS, the KD11D processor has issued NPG and the device has returned SACK L causing NO SACK TD 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 (Ø) L - When this input is true, all of the above signals are overridden. Generated on print K6, 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 to BBSY before asserting theirs. (DATIP operations dictate worst case bus latencies for NPR devices).

BUS SSYN L - Another data transfer is still being completed and therefore the processor must wait before initializing another.

If none of the above BUS IN USE conditions exist, the K1Ø DAT TRAN (1) H signal clears the E121 flip-flop and activates K6 START TRAN H (start transfer). The RC circuit on the output of E121 illiminates any noise that may result from the synchronizer under worst case conditions.

5.7.2.3 Bus Control

Once the K6 START TRAN H signal is activated, the DAT TRAN circuitry begins a UNIBUS data transfer operation by asserting K6 ASSERT ADDRESS L. As shown in the logic diagram of Figure 18, K6 ASSERT ADDRESS L triggers the following bus actions.

1. Enables the BUS ADDRESS (BUS AØØ-BUS A15) drivers (print K1 thru K4).

2. Enables the BUS BBSY driver (print K6).

3. Enables the bus control signals BUS CØ and BUS C1 which determine the type of transfer being performed.

C1	CØ	OPERATION
Ø	Ø	DATI
Ø	1	DATIP
1	Ø	DATIO
1	1	DATOB

The actual condition of these control lines is determined by the CONTROL STORE outputs K1Ø CØ (1) H and K1Ø C1 (1) H.

4. Enables the BUS DATA (BUS DØØ-BUS D15) drivers if the operation being performed is a DATO.

Figure 18 DATA TRANSFER BUS CONTROL

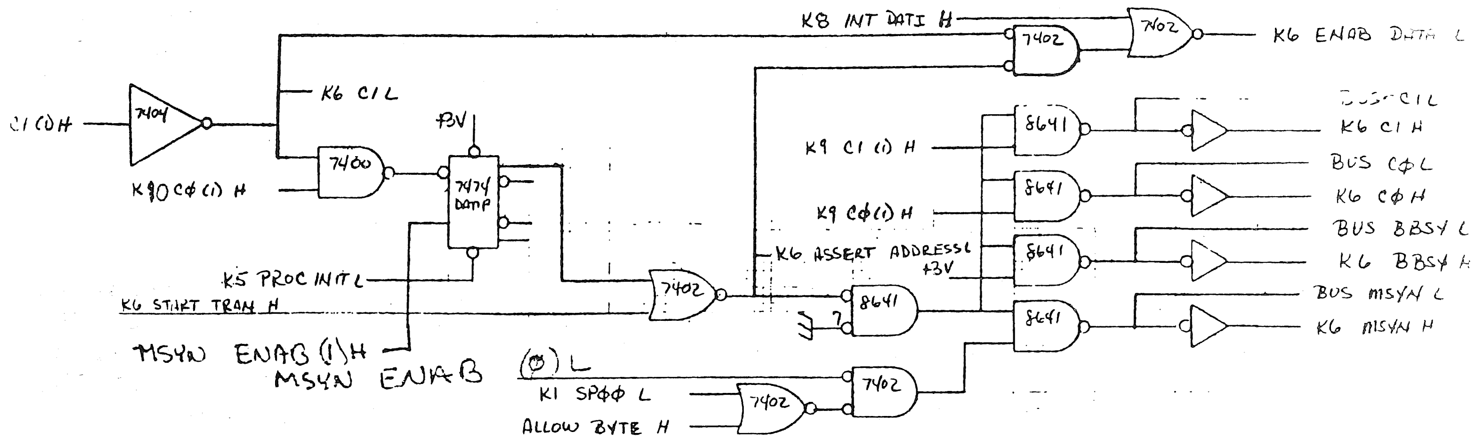
5.7.2.4 MSYN/SSYN TIMEOUT Circuitry

UNIBUS specifications require that the BUS MSYN L control signal be enabled no sooner than 15Ø ns after the bus address, data, and control lines have been asserted. To meet this requirement, the circuitry in Figure 19 has been incorporated into the DAT TRAN logic (print K6).

Figure 19 MSYN/SSYN CONTROL

The first one-shot, E98, delays the triggering of the SSYN TIMEOUT

Page 76A



DATA TRANSFER BUS CONTROL
FIGURE 18

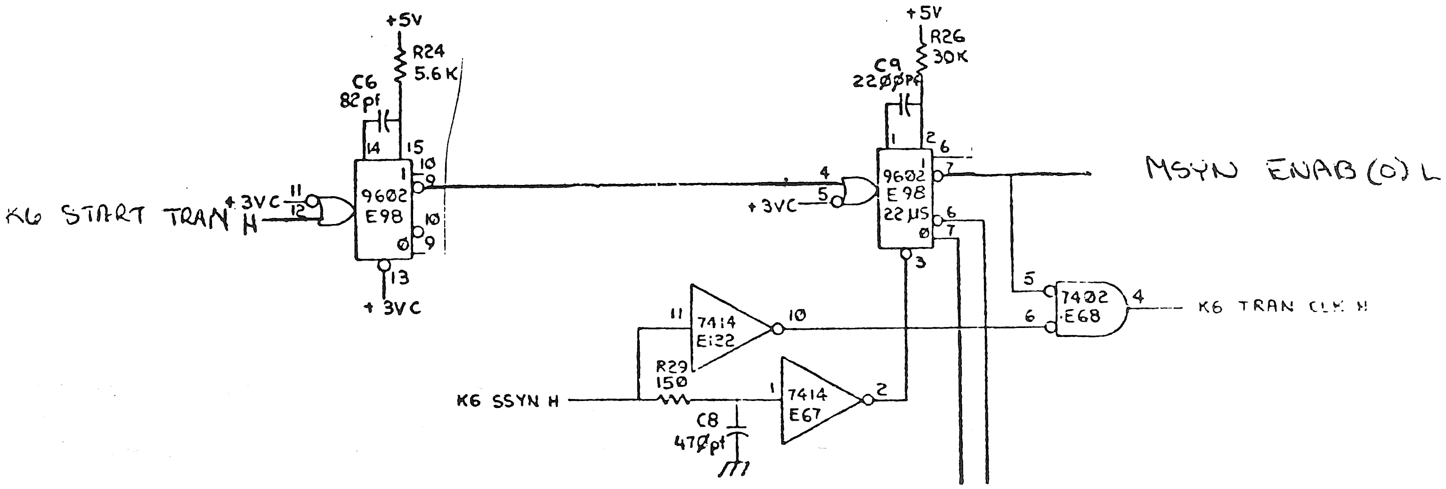


FIGURE 19

MSYN/SSYN CONTROL

one-shot (E98) until approximately 250 ns after the assertion of K6 START TRAN H. Once fired, the output of the SSYN TIMEOUT one-shot enables the BUS MSYN L bus driver and waits for the bus peripheral being accessed to return a BUS SSYN L. When BUS SSYN L is returned, E98 is cleared 75 ns after the SSYN is received negating MSYN and clocking data obtained from memory into the BREG or INSTRUCTION REGISTER.

5.7.2.5 BUS Errors

Once the SSYN TIMEOUT one-shot is triggered, SSYN must be returned within 22 microseconds. If SSYN is not returned in this time, E98 times out setting the BUS ERROR (BE) flip-flop E115. Upon entering the next SERVICE microcode state, the processor will monitor the status of the BE flip-flop and trap if the BE flip-flop is set.

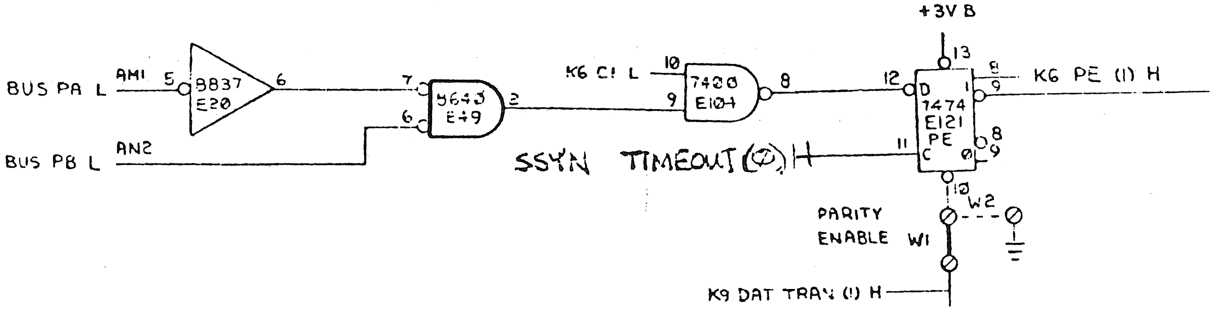
5.7.2.6 PARITY Errors

Along with clocking data into the BREG, IR, and BE latch, the timeout of E98 also clocks. The parity error detection logic shown in Figure 20.

Figure 20 PARITY ERROR CIRCUIT

If a data transfer is being performed with a parity memory option (MS11-FP, MS11-HP, MM11-CP or MM11-DP) all parity errors detected by the memory will be reflected back to the KD11D or the UNIBUS lines BUS PA L and BUS PB L.

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



PARITY ERROR CIRCUIT
FIGURE 20

Errors found while performing a DATIP or DATI (K6 C1 L is true) will result in the PARITY ERROR flip-flop (E121) being set when E98 times out. Processor operations resulting from PARITY ERROS will be discussed further in the BUT SERVICE section to follow.

Note that the entire PARITY ERROR circuit can be disabled by removing jumper W1 and inserting another jumper in the space provided for jumper W2. Note also, that the detection of a PARITY ERROR forces a BUS ERROR conditon.

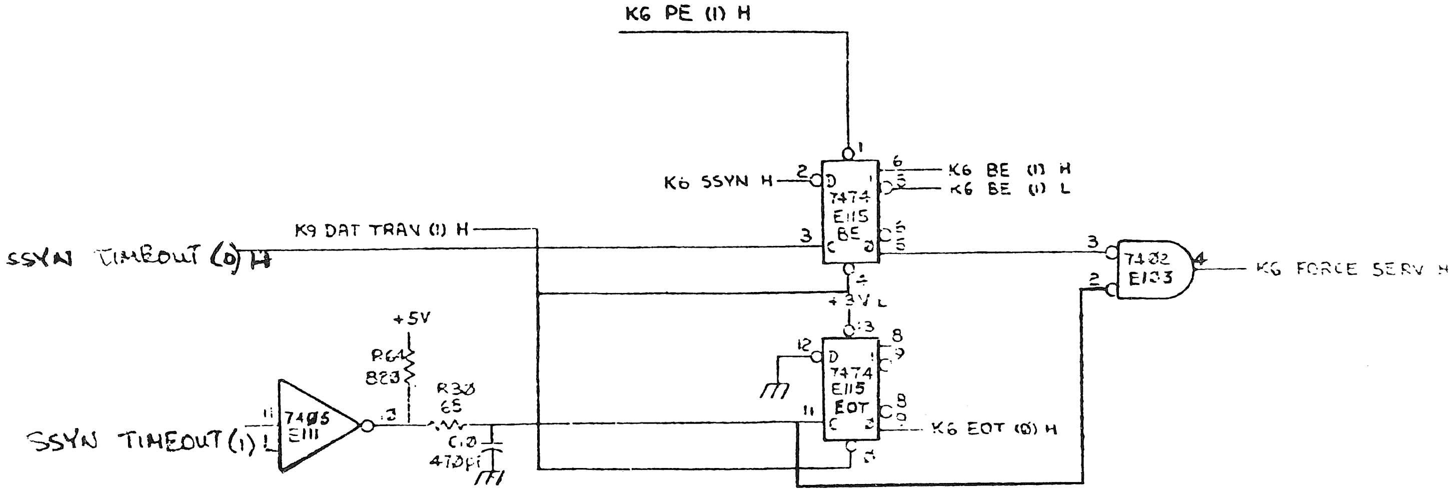
5.7.2.7 End of Transfer Circuitry

To synchronize the DAT TRAN logic with the main KD11D processor clock, the END OF TRANSFER (EOT) circuitry has been incorporated into the CPU (print K6). Approximately 100 ns after the SSYN TIMEOUT one-shot (E98) times out, the EOT flip-flop (E115) is clocked removing the previously discussed processor clock disabling signal K6 TRAN INH L. If a BUS ERROR has been detected, the delayed signal that clocked the EOT flip-flop generates a 100 ns pulse on the K6 FORCE SERV H line. This pulse clears the micro-pc address latches (MPC00-MPC07) on print K9 forcing the processor to enter the SERVICE microroutine on the next PROC CLK L low-to-high transition. An explaination of the terms micro-pc and microroutine is available in the CONTROL STORE section which follows later.

Figure 21 END-OF-TRANSFER CIRCUITRY

5.7.2.8 Data-In-Pause Transfer

Another circuit included in the DAT TRAN logic detect DATA-IN-PAUSE (DATIP) transfers and controls the bus control signal BBSY. Upon initiating a DATIP (READ-MODIFY-WRITE) bus operation, the flip-flop



END-OF-TRANSFER CIRCUITRY
FIGURE 21

E97 is latched forcing the processor to hold K6 BUS 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.

Figure 22 DATA-IN-PAUSE CIRCUITRY

5.7.2.9 Odd Address Detection

To prevent odd addressing errors, two NOR gates (E68) have been inserted between the SSYN TIMEOUT one-shot (E98) and the BUS MSYN driver. These gates prevent the assertion of MSYN if an odd bus address is being placed on the UNIBUS (K1 SP00 L is true) without the approval of the microroutine being performed (CONTROL STORE output K10 ALLOW BYTE H true). If this condition exists, the SSYN TIMEOUT one-shot would be allowed to timeout without ever asserting BUS MSYN L and thus never receiving BUS SSYN. The end result of this operation would be the detection of a BUS ERROR.

5.8 Power Fail/Auto Restart

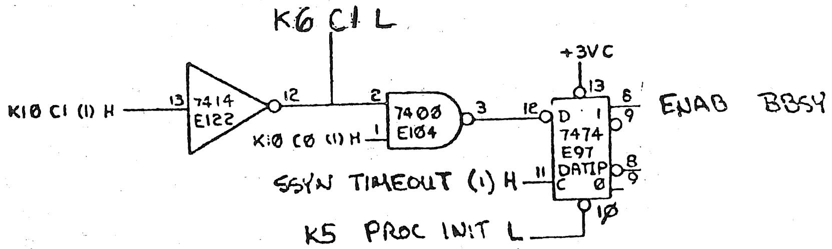
The KD11D power fail/auto restart circuitry (print K5) serves the following purposes:

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 properites of the Unibus drivers and



DATA-IN-PAUSE CIRCUITRY
FIGURE 22

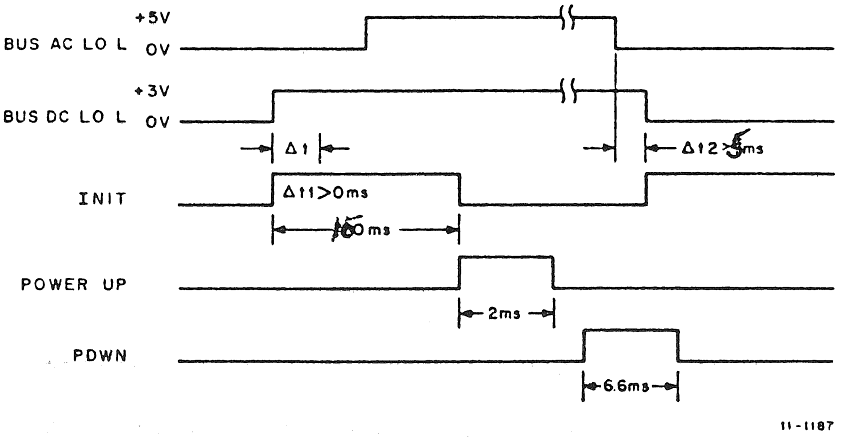


FIGURE 23
~~Figure 8-19~~ BUS AC LO and BUS DC LO Timing Diagram

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.

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 (Figure 23). The power-up sequence 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.

Figure 23 BUS AC LO and BUS DC LO Timing Diagram

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 DEC use BUS DC LO L as a switch signal, turning them off, even if power is still available. Time △ +2 (Figure 23) is the time delay between the assertion of BUS AC LO L and the assertion of BUS DC LO L, it 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 equipment, △ +2 must be greater than 5 ms.

When a pending power fail is sensed, a program trap occurs causing the present contents of R7 and the PSW to be pushed onto the memory stack, as determined by the contents of R6 (Stack pointer register). R7 is then loaded with the contents of memory location 24(8), the PSW is loaded with the contents of location 26(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 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 found on print K5. One-shot E110 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, E103, is fired if BUS AC LO L is not asserted and the processor begins the R7 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 presets the POWER INIT flip-flop E109. Setting this latch 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 been reset, the assertion of BUS AC LO L fires the one-shot, PDWN, E103. Flip-flop E97 is set 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 ROM E71 (print K8).

If a momentary power failure occurs which 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 (0) L one-shot times out, retriggering the INIT one-shot simultaneously with DC LO H becoming nonasserted.

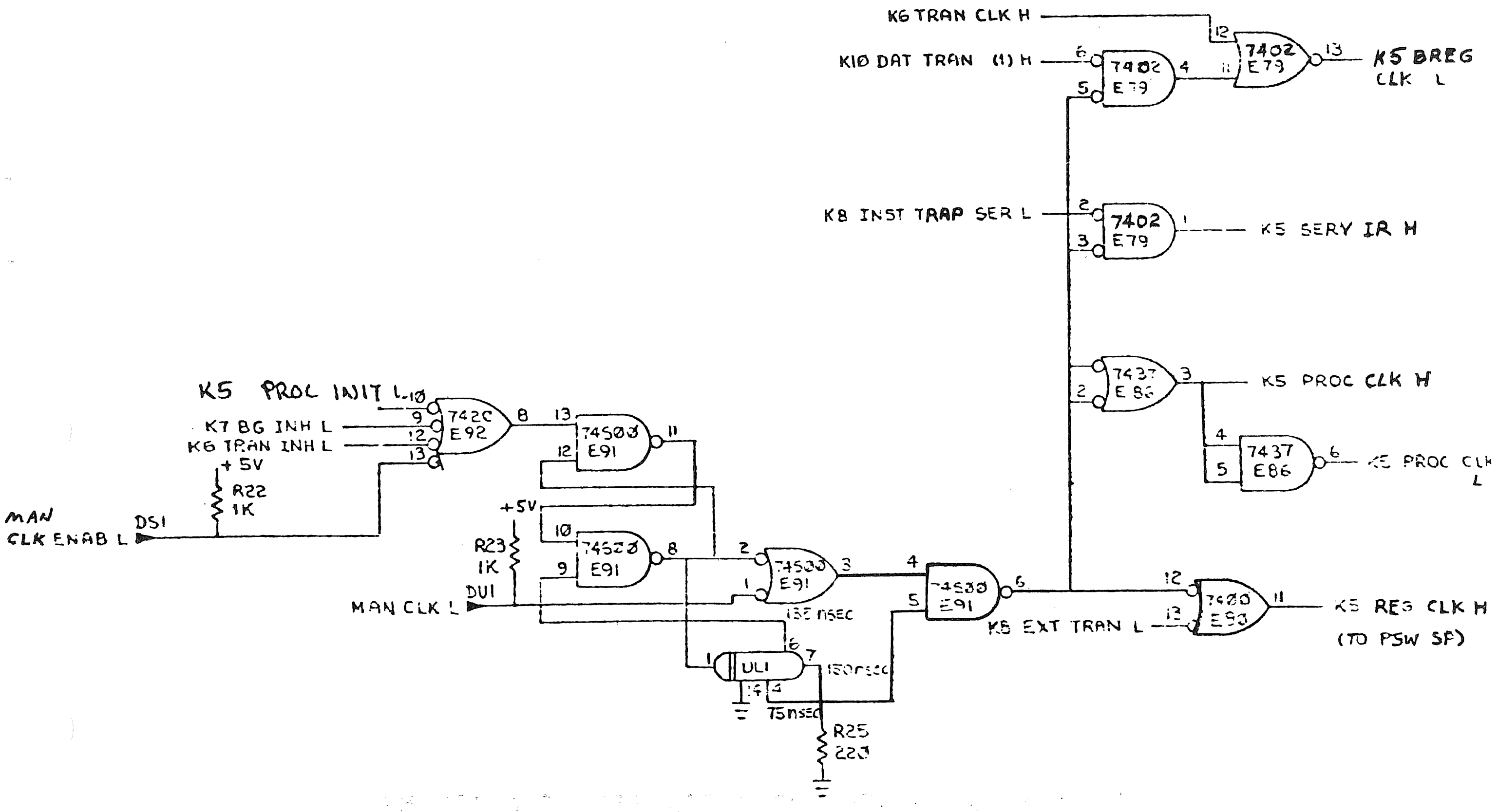
When a RESET instruction is decoded by ROM E76, the ROM output signal K12 START RESET L is clocked into the START RESET flip-flop E109 (print K5). This flip-flop output triggers a 100 ms INIT, afterwhich the processor continues operation.

5.9 PROCESSOR CLOCK

The KD11D processor clock circuitry is shown in Figure 24 and on print K5. A single delay line is used to generate a pulse train to which the entire processor is synchronized. Since it is a fully clocked processor, events that result in the alteration of storage registers occur only on defined edges of the processor clock.

Figure 24 PROCESSOR CLOCK

If all clock disable inputs are unasserted, the clock will begin running as soon as +5 volts is applied. The period of the oscillator pulse output is fixed at 260 ns as per Figure 25.



PROCESSOR CLOCK
FIGURE 24

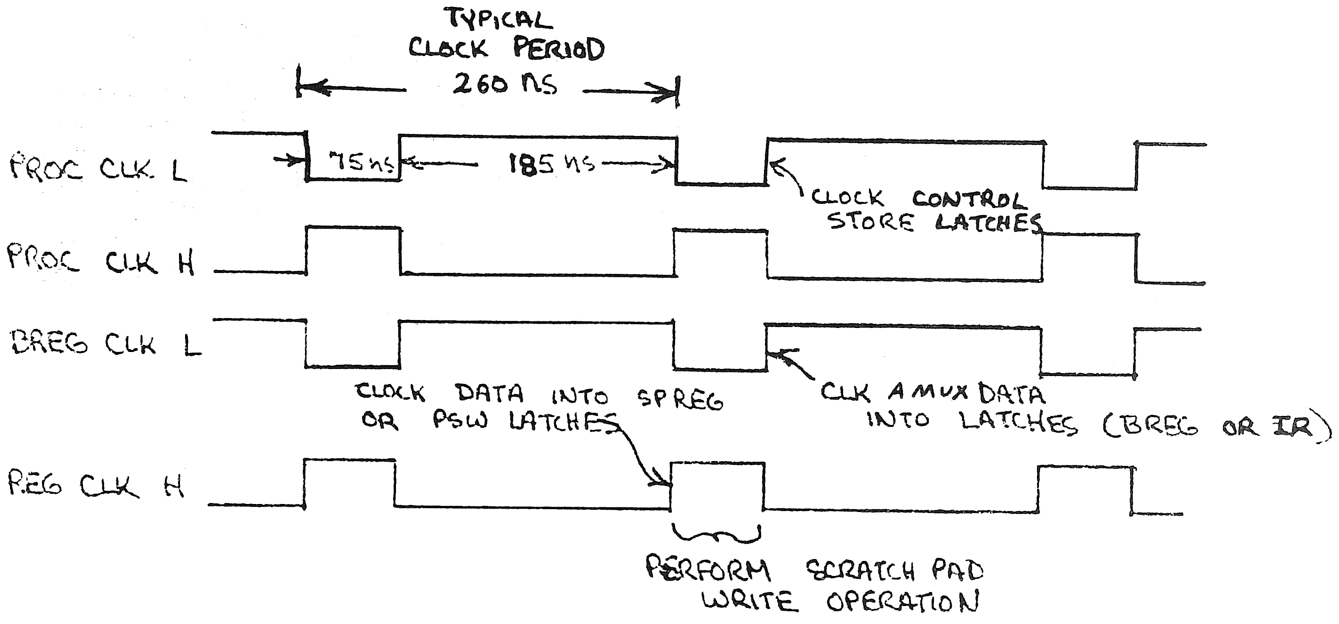
Figure 25 PROCESSOR CLOCK TIMING DIAGRAMS

The clock is turned on and off by means of gating the feedback through its delay line. It is turned off under the following conditions by the appropriate signal:

1. During a BUS INIT from another device.
2. The INIT portion of power up routine.
3. The INIT portion of 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.
8. During bus data transfers.
9. After executing a HALT instruction.
10. When the manual clock is enabled.

5.10 PRIORITY ARBITRATION

5.10.1 BUS Requests


TYPICAL CLOCK PERIOD
260 ns
75 ns
185 ns
PROC CLK L
CLOCK CONTROL STORE LATCHES
PROC CLK H
BREG CLK L
CLOCK DATA INTO SPREG OR PSW LATCHES
CLK AMUX DATA INTO LATCHES (BREG OR IR)
REG CLK H
PERFORM SCRATCH PAD WRITE OPERATION

PROCESSOR CLOCK TIMING DIAGRAMS
FIGURE 25

The KD11-D 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 four BR lines, numbered 4, 5, 6, and 7 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 bits (07-05) of the PSW, is at level 4, only devices that request BRs at levels higher than 4, such as BR 7, BR 6, or BR 5, are serviced. Table 13 contains the order of priorities for all BRs and other traps.

Priority	Service Order
Highest	HALT Instruction
	BUS ERRORS
	INSTRUCTION TRAPS
	TRACE TRAPS
	STACK OVERFLOW
	POWER FAIL
	HALT SWITCH
	BR7
	BR6
	BR5
	BR4
Lowest	Next instruction fetch

PRIORITY SERVICE ORDER
TABLE 13

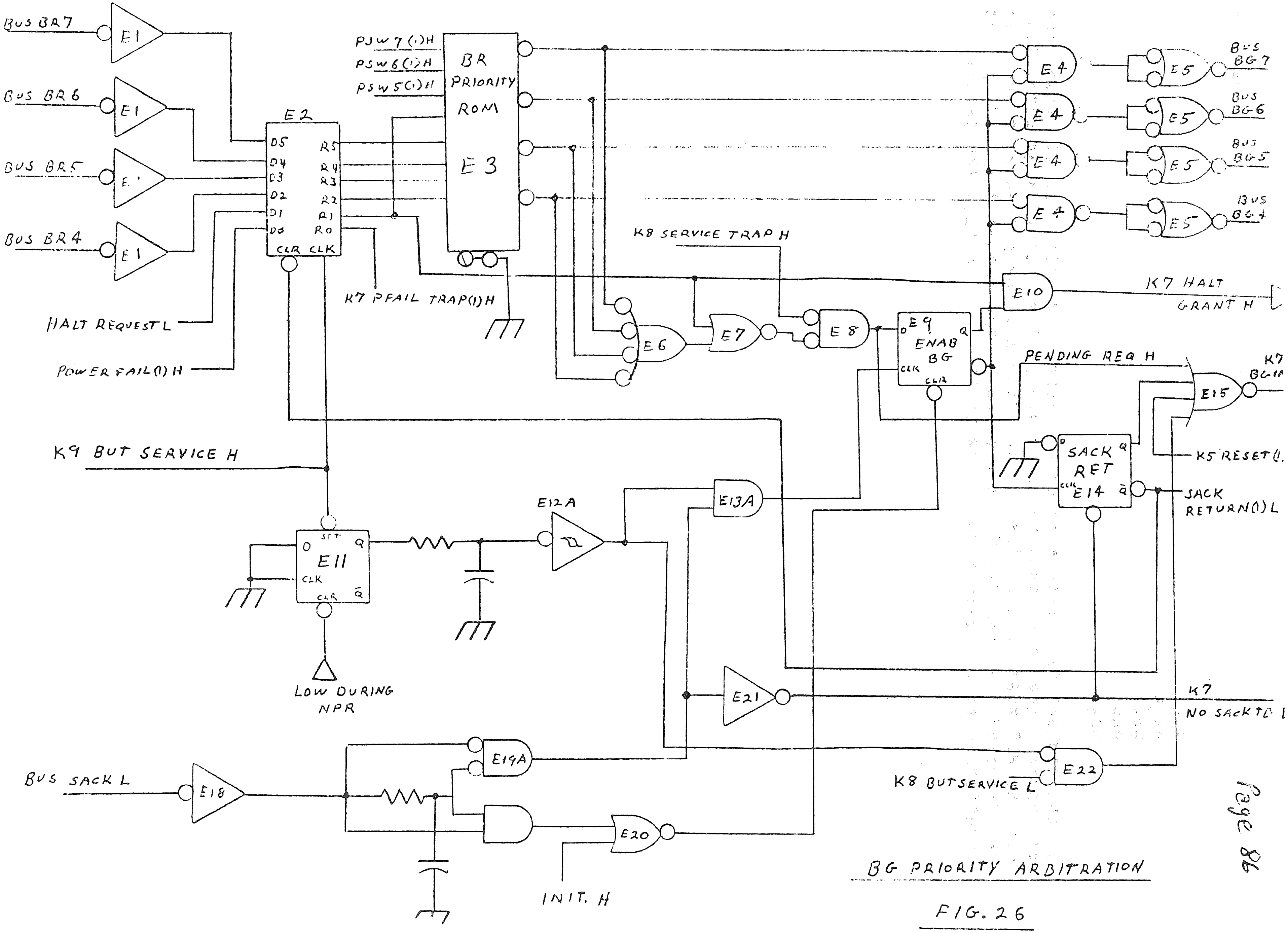
Since a BR can cause a program interrupt, it may be serviced only after the completion of the current instruction in the IR. A device that requests a program interrupt must at the appropriate time place a vector address on the Unibus data lines. The processor first stacks away the current contents of PSW and R7; then a new R7 is loaded from the contents of the vector address, and a new PSW is loaded from the contents of the vector address plus two. Further descriptions of how the processor handles this BR routine will be discussed in the SERVICE section to follow.

Arbitration logic for BRs is shown on print K7 and in Figure 26. All BRs are received directly from the UNIBUS (UNIBUS receivers E20, and E32) and latched into register E14 (74174 Quad D-Type latch) when the microprogram enters the next SERVICE state (K9 BUT SERVICE (1) H is true). The BR PRIORITY ABRITRATION ROM (E7) 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 E7 in the order of priority are shown below:

HLT RQST
PSW7
BR7
PSW6

BR6
PSW5
BR5
PSW4
BR4

If the highest BR received is of a higher priority level than the processor, the corresponding grant enable ROM output is asserted low. With no HLT RQST or trap instruction pending, the processor clock will be disabled by the K7 BG INH L signal. The actual BUS GRANT is not transferred to the UNIBUS until the ENABLE BG flip-flop E55 is set.



BG PRIORITY ARBITRATION

FIG. 26

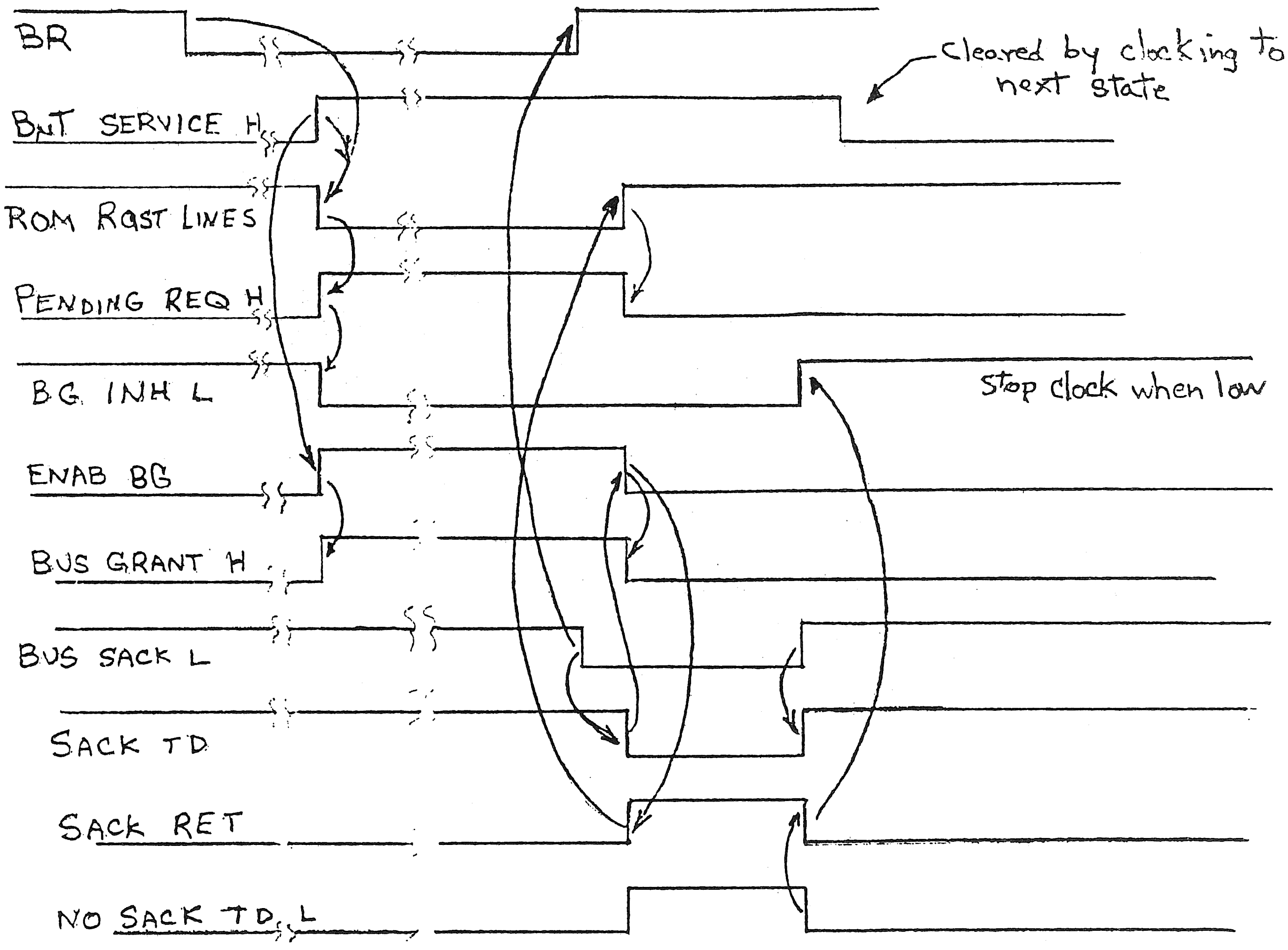
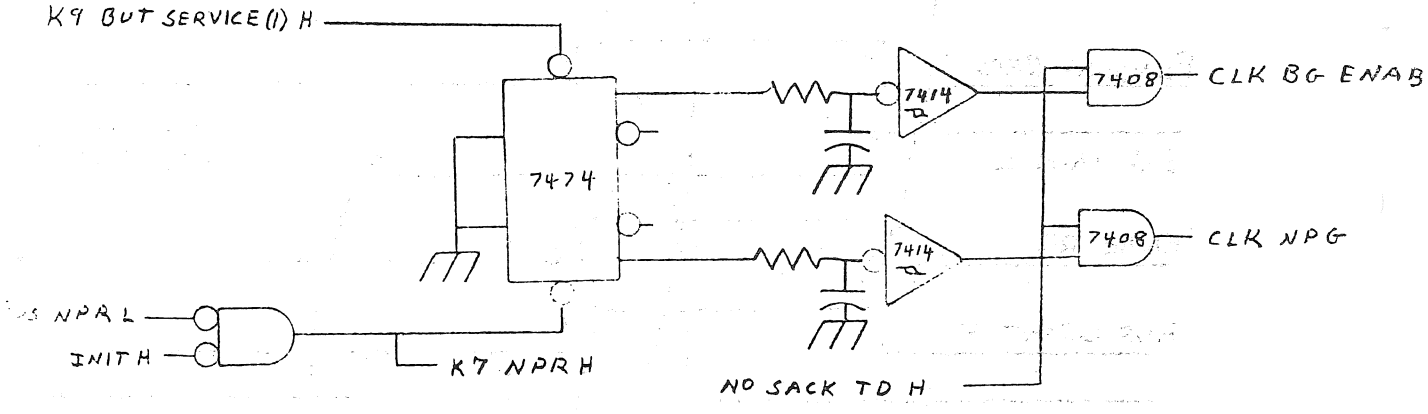


Figure 27 BUS REQUEST TIMING

Grants both BG and NPG are controlled by the synchronizer logic shown below and on print K7.

Figure 28 GRANT SYNCHRONIZER



GRANT SYNCHRONIZER

FIGURE 28

This circuitry arbitrates whether a BG or an NPG (Non-Processor Request Granted) will result depending on which flip-flop input line (set or reset) was deactivated first. If the set input K9 BUT SERVICE (1) H is detected first, the Q output of E73 (pin 9) will transition low. After a delay of 175 ns, this signal will clock the ENAB BG flip-flop E55 provided there is no BUS SACK L signal on the UNIBUS. Once E55 is set the bus grant arbitrated by ROM E7 is channeled onto the UNIBUS (bus drivers E26). Once the requesting peripheral receives BG, it then returns BUS SACK L.

Upon receiving BUS SACK L, the processor then clears its ENAB BG flip-flop removing the BUS GRANT from the UNIBUS and sets the SACK RETURN flip-flop to keep the processor clock disabled.

Removal of BUS GRANT causes the peripheral to drop its BUS SACK L, assert BUS INTR L and enable a vector address onto the UNIBUS data lines. The processor then deskews the removal of SACK, clears the SACK RET flip-flop (E73) and enables the processor clock again. Once in operation, the processor clocks the peripheral vector address into the BREG, returns BUS SSYN L and begins running the microcode trap routine which branches the processor to the interrupt handling program determined by the vector obtained.

5.10.2 Non-Processor Requests (NPR)

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 simply to give up 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 E73, K7 NPR H becomes unasserted before the set input, the Q output will transition low causing the NPG flip-flop E55 to be set if BUS SACK L is not true. The output of this flip-flop enables the BUS NPG H Unibus line granting the bus to the Non-Processor device. The requesting device will then return BUS SACK L clearing the NPG and will wait until the bus is free (no BBSY).

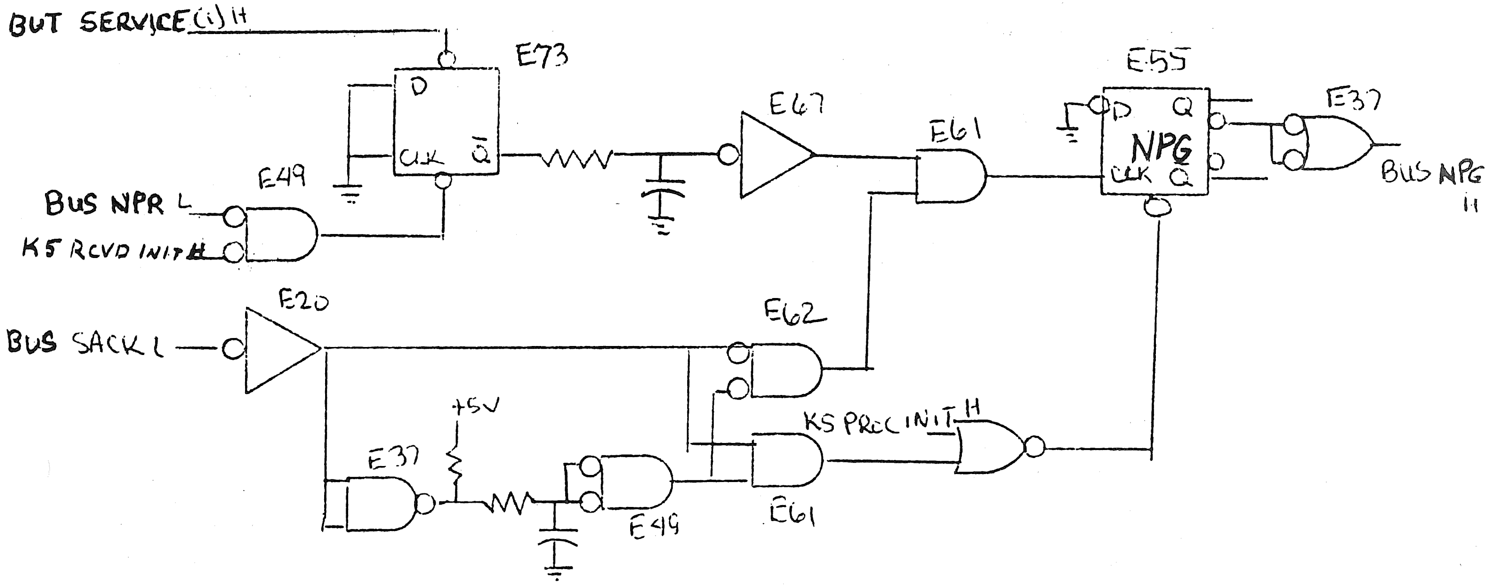
Figure 29 NPG PRIORITY ARBITRATION

5.1Ø.3 Halt Grant Requests

Unlike all previous PDP-11 processors the KD11D has implemented what could be considered another priority level; K12 HLT RQST L. This input is used to monitor the USER'S CONSOLE HALT/CONTINUE switch. If a HALT is detected (K12 HLT RQST L activated), the processor will recognize it as an interrupt request (priority level is shown in Figure 13) upon entering the next SERVICE microstate. The processor will then inhibit the processor clock (Figure 26) and return a recognition signal (K7 HLT GRANT H). Upon receiving K7 HLT GRANT H, the console drops the K12 HLT RQST L and asserts BUS SACK L gaining complete control of both the UNIBUS and KD11D.

The User can maintain the processor in this inactive (HALTED) state indefinitely. Upon releasing the HALT switch, the Users Console releases BUS SACK L and the processor continues operation as if nothing had happened.

5.11 SERVICE TRAPS



NPG PRIORITY ARBITRATION
FIGURE 29

5.11.1 General Description

All interrupts, error traps, and instruction traps are recognized and serviced by the KD11D when the processor enters what is called the SERVICE micro-instruction state. The functions performed during this state are most critical to the operation of the processor and should be completely understood.

Upon entering the SERVICE state, all bus interrupts, error traps, and instruction traps realized during the performance of the last instruction are arbitrated by the SERVICE ROM E71 print K8). Each trap condition is then serviced according to its priority as shown in Table 13.

5.11.2 Circuit Operation

Rom E71 services a specific trap by generating a vector address unique to that trap condition (Table 15). 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 which could correct or indicate that a specific error occured.

The various trap conditions which cause the processor to vector are as follows.

BUS ERRORS - A BUS ERROR indicates that the processor has attempted to access non-existent memory or a memory location that did not return BUS SSYN within 22 usec. The detection circuitry for bus errors was previously described in the DAT TRAN section.

Once detected, the bus error condition of flip-flop E115 (print K6) is clocked into latch E101 (print K10) on the next low-to-high transition of PROC CLK L creating the error signal K BE FLAG (1) H. Double buffering is required because E115 is cleared at the end of the data transfer micro-instruction step and used again to detect a Double Bus Error condition during the trap routine.

STACK OVERFLOW ERROR - Any attempt by the processor of decrementing the contents of the STACK POINTER REGISTER (R6) beyond the 400 location stack limit (K11 8-15=0 L) of the KD11D will result in the STACK OVERFLOW flip-flop E134 (print K8) being set on the next high-to-low transition of PROC CLK H.

Figure 3Ø STACK OVERFLOW

PARITY ERROR

- a PARITY ERROR indicates that the processor attempted to input data from a parity memory and that memory indicated a parity error.

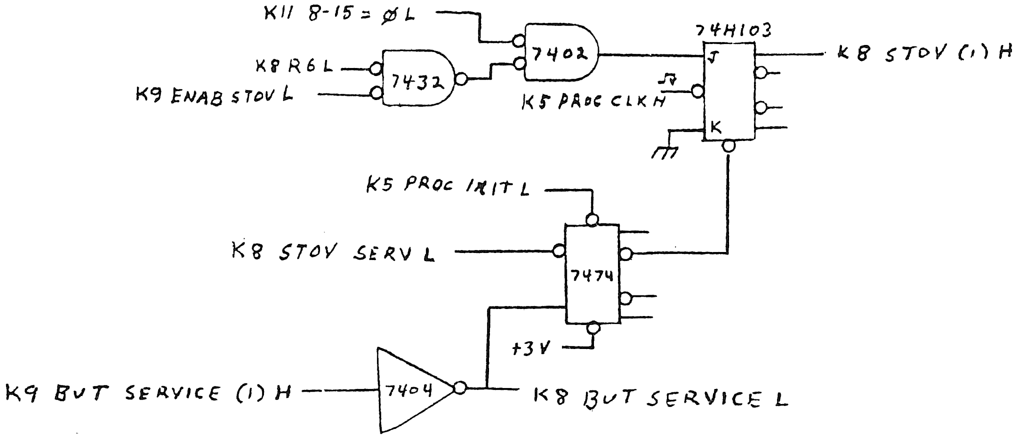
  The PARITY ERROR detection circuitry was previously described in the DAT TRAN section. Once detected the error condition of flip-flop E121 is clocked into latch E1Ø1 (print K1Ø) on the next PROC CLK L low-to-high transition creating K1Ø PE FLAG (1) H. Double buffering is necessary because E121 is cleared at the end of the data transfer microinstruction step allowing E115 to detect a double bus error condition.

POWER FAILURE

- The output of the PWR FAIL flip-flop E97 (print K5) is set when the power supply asserts BUS AC LO L indicating loss of AC power.

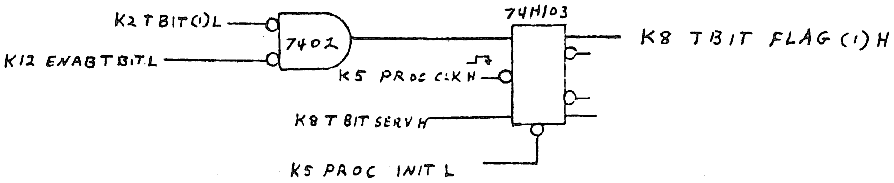
TRACE TRAP

- This trap is program controlled by the user allowing him to insert a processor/user interactive subroutine into his main program. The trap is enabled by setting the PSW TBIT (K2 TBIT (1) L). Upon completion of the next instruction (K12 ENAB TBIT L), the J-K flip-flop E134 is set creating the KT BIT FLAG (1) H signal.



STACK OVERFLOW

FIGURE 30



T BIT FLAG

FIGURE 31

Figure 31 TBIT FLAG

IR CODE ØØL-IR CODE Ø2L - These three binary coded trap signals are generated by the IR DECODE ROMS E69, E7Ø and E75 on print K12, and indicate the following trap conditions.

TRAP CONDITION	IR CODE LINES		
	Ø2	Ø1	ØØ
HLT INSTRUCTION	Ø	Ø	Ø
TRAP INSTRUCTION	Ø	Ø	1
EMT INSTRUCTION	Ø	1	Ø
IOT INSTRUCTION	Ø	1	1
BPT INSTRUCTION	1	Ø	Ø
ILLEGAL INSTRUCTION	1	Ø	1
RESERVED INSTRUCTION	1	1	Ø
UNUSED (none of above)	1	1	1

TABLE 14

Upon entering the SERVICE micro-instruction state, the SERVICE Rom E71 monitors any combination of the above trap conditions. If any inputs are enabled, the Rom forces the processor to branch to a special TRAP routine on the next micro step by asserting the micro-pc address line MPCØØ L. While still in the SERVICE state, the Rom also generates a specific vector address (Table 15) using outputs C2, C3 and C4 and channels it onto the processor AMUX lines by activating K9 AMUX SØ L where it is then latched in the BREG.

Before leaving the SERVICE state E71 also clears the condition which caused to original trap. This is done by asserting one of the following outputs; K8 TBIT SERV H, STOV SERV L, PFAIL SER L or INST TRAP SER L. The first three of these outputs clear their respective trap signals directly. For those traps specified by the IR CODE lines, however, it is necessary to remove the instruction in the IR. This operation is performed by the INST TRAP SER L output which ORs with the PROC CLK to generate K5 SERV IR L which in turn removes the trap instruction from the IR. This operation prevents the processor from looping on the same trap condition.

For BUS REQUEST (BRs), the BUS INTR control signal is allowed to force MPCØØ L during SERVICE provided there are no other traps of higher priority. By enabling this line the processor will branch to the TRAP ROUTINE and vector to the address specified by the BR device. If there is a trap of higher priority BR interrupts are prevented from receiving BG by the SERVICE TRAP L output of E71.

OCTAL UNIBUS VECTOR ADDRESS	TRAP CONDITIONS
004	Time-out & other error
010	Illegal & reserved instructions
014	BPT, breakpoint trap
020	IOT, input/output trap
024	Power Fail
030	EMT emulator Trap
034	TRAP instruction
114	Memory Parity Error

VECTOR ADDRESSES
TABLE 15

5.12 CONTROL STORE

5.12.1 General Description

The CONTROL STORE circuit (print K9 and K10) consists of five 256 word by 8 bit bipolar Roms, seven Quad D-Type flip-flops and an assortment of gates and multiplexers. This logic operates in a similar fashion to a microprocessor having eight address lines and 30 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 unts (ALU), influencing the DAT TRAN circuitry or in general controlling the total KD11D. The contents of each location is configured in a manner that allows sequences of locations to be combined into microroutines which perform the various PDP-11 instruction operations. Each Rom location is therefore considered a microinstruction or microstep.

5.12.2 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 output of Rom E138 is loaded into the MPC (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 are as follows.

Instruction Decode - As previously mentioned, the microroutines contained in the CONTROL STORE are designed to efficiently perform the operations specified by the various PDP-11 instructions. Specific microroutines are implemented for specific instructions. The main purpose for the IR DECODE circuitry is to translate the PDP11 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. An adequate description of the specific addresses for each instruction was included in the IR DECODE section.

TRAP DECODE - Routines have also been included in the CONTROL STORE to implement error routines which push and pop the PC and PSW onto or off the processor stack. Upon request of the CONTROL STORE (BUT SERVICE (1) H), the MPC ØØ line can be enabled by the SERVICE ROM (E71) causing a microbranch to one of these microroutines.

BRANCH ON BYTE - The various PDP-11 instructions are dependent on whether an even or odd byte operation is being performed. Modifications to the sequence of microsteps used for specific instructions can be made by enabling (BUT BYTE L) microbranches based on even or odd byte instructions in the IR.

PWR RESTART - Upon performing a power restart, the MPC is cleared by INIALIZE (INIT). The PWRUP circuitry on print K5 then enables the MPC ØØ line (PWR INIT flip-flop E1Ø9) forcing the CONTROL STORE to perform the PWR UP routine beginning at MPC address one (ØØ1).

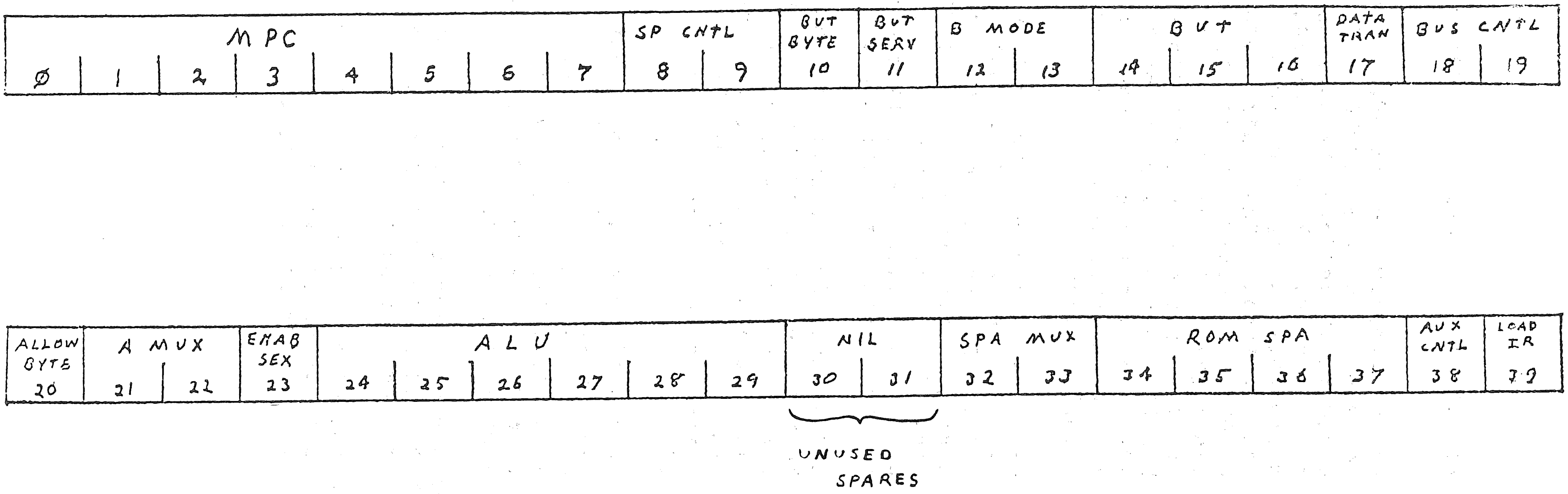
In general, microsteps are not executed from numerically sequential locations in the CONTROL STORE and care should therefore be taken in following the flows described in Chapter 4.

Figure 32 shows the format of all 256 words in the KD11D CONTROL STORE. The fields, the possible values they contain, and the significance of each value are described below.

Page 95

11/04

CONTROL STORE ROM



MPC								SP CNTL		BUT BYTE	BUT SERV	B MODE		BUT			DATA TRAN	BUS CNTL	
Ø	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

ALLOW BYTE	A MUX		ENAB SEX	ALU						NIL		SPA MUX		ROM SPA				AUX CNTL	LOAD IR
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

$\underbrace{30 \quad 31}_{\text{UNUSED SPARES}}$

FIGURE 32 KD11-D Microinstruction Format (ROM

5.12.3 CONTROL STORE FIELDS

FIELD	FIELD LENGTH	DESCRIPTION
MPC	8	Eight bit micro-pc address which specifies the ROM location of the next microstep to be performed.
SP CONTROL	2	Determines scratch pad operations according to the following format.

SP CONTROL 01	SP CONTROL 00	OPERATION
0	0	Read
0	1	Write low byte
1	0	Write word and enable ENAB H
1	1	Write word.

FIELD	FIELD LENGTH	DESCRIPTION
BUT BYTE	1	Allows IR DECODE logic to force an MPC branch if the instruction being performed is a byte. Branches will be a follows. Even Byte +2 Odd Byte +3 Actual branch logic is shown on print K12.
BUT SERVICE	1	Indicates that the processor has entered the SERVICE microstep. Enables the SERVICE ROM E71, causing the processor to recognize any pending errors or interrupts.
BMODE	2	Controls the operation of the B-Register during each microstep. The latched outputs of this field can be wire-ORed by other CPU logic. Coding of these signals is as follows.

BMODE 01	BMODE 00	OPERATION
0	0	HOLD DATA
0	1	SHIFT RIGHT
1	0	SHIFT LEFT
1	1	PARALLEL LOAD

FIELD	FIELD LENGTH	DESCRIPTION
BUT	3	Multiplexed control lines which generate the following enable signals.

BUT DEST L - Enables microbranch to destination operand microcode sequence. Corresponding logic is on print K9.

ENAB STOV L - Enables stack overflow detection circuit on print K8.

ENAB DBE L - Enables circuitry which forces processor to halt on detecting a bus error during this microstep. Corresponding logic on print K11.

LOAD PSW L - Allows the PSW register to be loaded upon completion of this microstep. See prints K1 and K2.

LOAD CC L - Allows the condition codes N, Z, V and C to be loaded upon completion of this microstep. Circuitry is shown on print K1.

BUT 02	BUT 01	BUT ØØ	OPERATION
Ø	Ø	Ø	UNUSED
Ø	Ø	1	UNUSED
Ø	1	Ø	BUT DEST L
Ø	1	1	LOAD CC L
1	Ø	Ø	ENAB DBE L
1	Ø	1	LOAD PSW L
1	1	Ø	ENAB STOV L
1	1	1	UNUSED

DAT TRAN (1) H 1 Enables data transfer circuitry on print K6. Indicates that the processor is performing a UNIBUS transfer during this microstep.

BUS CONTROL 2 Enables the UNIBUS control lines BUS CØ L and BUS C1 L as follows.

C1(1)H	CØ(1)H	TRANSFER
Ø	Ø	DATI
Ø	1	DATIP
1	Ø	DATO
1	1	DATOB

ALLOW BYTE H 1 Gates the UNIBUS control BUS MSYN L when byte instructions are being performed (print K6). Also helps generate the signal K8 INH +1 L during byte operations.

AMUX 2 Controls the select lines of the AMUX according to the following.

AMUX S1	AMUX SØ	DATA

0	0	PSW
0	1	ALU outputs
1	0	Service vector
1	1	UNIBUS DATA

ENAB SEX (1) L	1	Enables the Data Path (prints K1 and K2) logic which extends the sign of the data in the low byte of the BREG (bit 07) through the bits of the upper byte.
ALU S3-ALU S0, ALU MODE, and ALU CIN	6	Determine the operation performed by the 16-bit ALU according to Table 9. These lines are also wire-ORed allowing the AUXILIARY CONTROL circuitry to determine the ALU operations according to Table 12.
SPA MUX	2	Controls the select lines of the Sratch Pad Address Multiplexer.

SPA MUX S1	SPA MUX S0	SPA OUTPUT
0	0	BUS ADDRESS BITS 00-03
0	1	INSTRUCTION REGISTER BITS 06-08
1	0	INSTRUCTION REGISTER BITS 00-02
1	1	CONTROL STORE ROM SPA 00-03

ROM SPA	4	Allow microinstructions from the CONTROL STORE to determine which Scratch Pad register will be addressed during the next microstep unless otherwise expressed by the SPA MUX control lines previously mentioned.
AUX SETUP	1	Enables the AUXILIARY CONTROL Roms during operate microinstructions.
LOAD IR	1	Allows loading of the Instruction Register. (Print K11)

6.0 MICROCODE

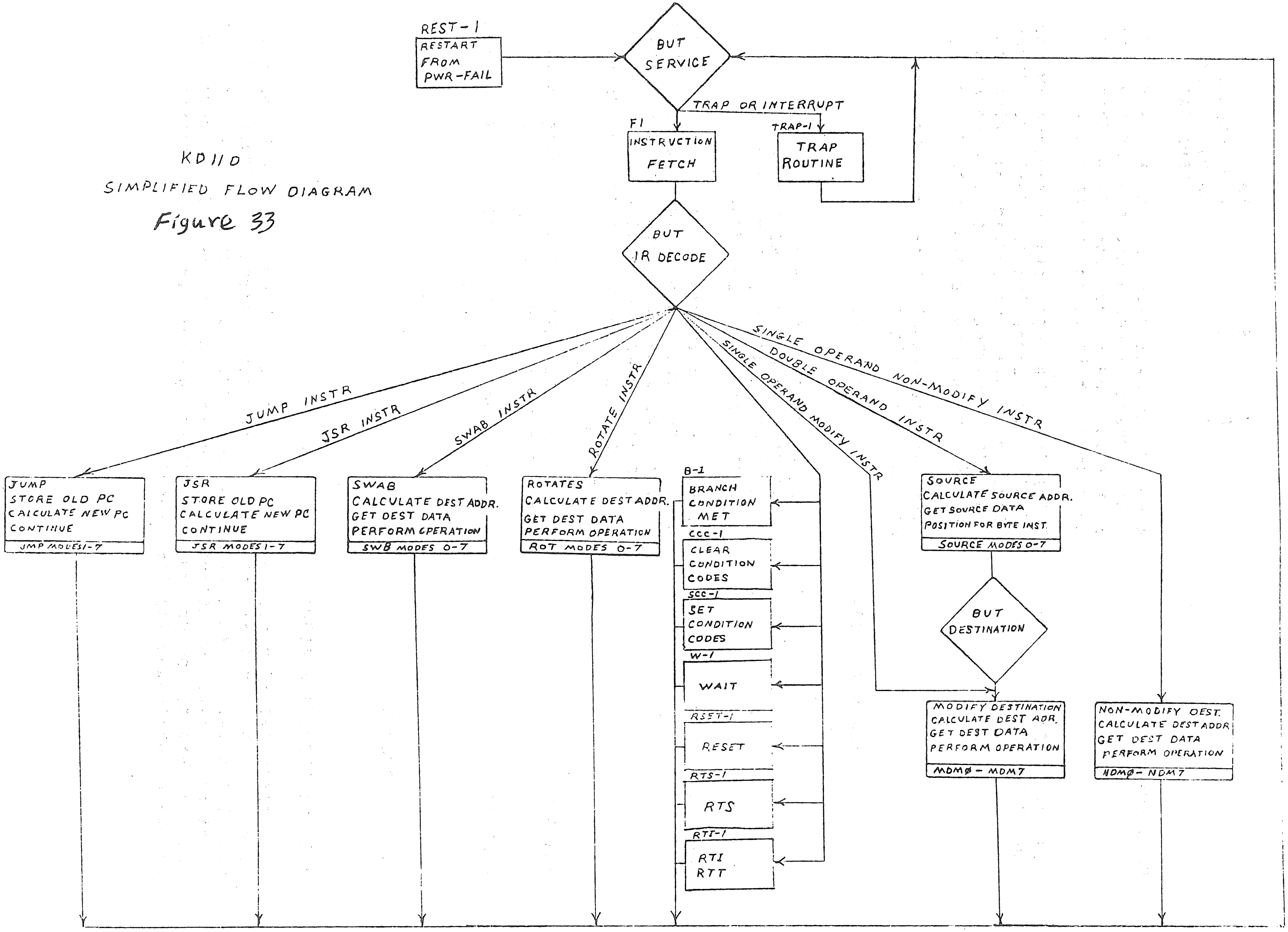
6.1 MICROPROGRAM FLOWS

A complete set of microinstruction flows is shown in block diagram form in the engineering circuit schematic package. Figure 33 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.

KD11D

SIMPLIFIED FLOW DIAGRAM

Figure 33



6.2 FLOW NOTATION

6.2.1 Microstep Mnemonic Names

All microsteps have mnemonic names which signify the type instruction being performed. A microroutine will often weave back and reuse part of another if the operations are identical. To understand the significance of the various mnemonic names the following definitions apply. All Xs shown indicate the instruction mode and Y indicate the step number.

MNEMONIC	DEFINITION
SMX-Y	Source mode microsteps for any double operand instruction.
MDMX-Y	Destination mode microsteps for Modifying double and single operand instructions.
NDMX-Y	Destination mode microsteps for Non Modifying double and single operand instructions.
FY	Microsteps contained in FETCH microroutine.
SERV	SERVICE microcode state
TRAP-Y	Microsteps used upon recognition of an interrupt, or trap instruction (IOT, BPT, EMT or TRAP).
ROTX-Y	Microsteps for ROTATE and Arithmetic Shift instructions.
SWBX-Y	Microsteps for SWAB instructions
JMPX-Y	Microsteps for JUMP instructions
JSRX-Y	Microsteps for JUMP to subroutine instructions.
RTS-Y	Microsteps for Return from subroutine instructions
RTI-Y	Microsteps for Return from Interrupt RTI and Return from TRAP (RTI) Instructions.
W-Y	Microsteps for WAIT instructions
RSET-Y	RESET instruction microsteps
CCC -Y	CLEAR Condition Code microsteps
SCC-Y	Set Condition Code microsteps
REST-Y	Microstep for restarting from a power failure.

SBE-Y   Source routine microsteps for even byte (except source mode 2) instructions.

SBO-Y   Source routine microsteps for odd byte (except source mode 2) instructions.

SMBE-Y   Source routine microsteps for source mode 2 even byte instructions.

SMBO-Y   Source routine microsteps for source mode 2 odd byte instructions.

MDB-Y   Modify byte instruction microsteps for destination mode Ø.

MXOB-Y   Modify odd byte instruction microsteps for destination mode X as shown.

MXEB-Y   Modify even byte instruction microsteps for destination mode X as shown.

NDEB-Y   Non-Modify even byte instruction microsteps for destination modes Ø and 1.

NDOB-Y   Non-Modify odd byte instruction microsteps for destination mode 1.

NMOB-Y   Non-Modify odd byte instruction microsteps for destination mode 2.

NMEB-Y   Non-Modify even byte instruction microsteps for destination mode 2.

MOV-Y   Microsteps for MOVE instruction destination mode Ø.

ROTB-Y   ROTATE byte instruction microsteps for destination mode Ø.

ROBX-Y   ROTATE odd byte instruction microsteps for destination mode X.

REBX-Y   ROTATE even byte instruction microsteps for destination mode X.

6.2.2 Flow Notation Glossary

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

FLOW NOTATION GLOSSARY

Designation	Definition
BA	Unibus Bus Address lines
-	Assignment operator
;	Separator
DATI	Initiate DATI operation on Unibus
Plus	Plus, the arithmetic operator
PC	Program Counter = scratch pad register 7 (R7).
B	B Register
IR	Instruction register
B SEX	B Reg sign extended (bit 7 repeated in bits 8 through 15).
RS	Scratch Pad Register specified by the source portion of the current instruction [IR (8;6)]
RD	Scratch Pad Register specified by the destination portion of the current instruction IR (2;Ø)]
Rn	Scratch Pad Register n specified by the CONTROL STORE ROM SPA lines.
ALBYT	Allow byte Unibus reference
ENAB OVER	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
RDB	Lower byte of the Scratch Pad Register specified by the destination portion of the current instruction.
J/m	Specifies m as the mnemonic of the next microstep.
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.
COND CODES	Set condition codes (N,Z,V and C) according to result of operation being performed by the ALU.
UNIBUS DATA	Data being received from the UNIBUS data lines BUS DØØ L-BUS D15 L.
B(SWAB)	Contents of B Register with upper and lower bytes swapped.
MINUS	MINUS the arithmetic operator.

6.3 MICROPROGRAM EXAMPLES

6.3.1 PDP-11 Instruction Interpretation

To illustrate the interpretation of PDP-11 instructions, the execution of a CMP instruction is traced through the microcode. The machine is in the RUN state (i.e., the machine is executing instructions) and the instruction is located in memory location 1ØØØ.

```
Location   Assembler Symbolic    Octal

  1000        CMP #15, CHAR      022767
  1002                           000015
  1004                           000100
    .
    .
    .
  1106        CHAR:  WORD 0
```

This instruction compares the literal 15 to the contents of CHAR and sets the condition code accordingly. Source mode is immediate (mode 2, register 7 = PC) and destination mode is relative (mode 6, register 7 = PC). Figure 34 shows the simplified flow for the CMP example.

Figure 34 CMP #15, CHAR (022767), Simplified Flow Diagram

First the instruction is fetched from memory (microsteps F1 and F2). This is the same fetch microroutine used to get each instruction from memory and update the PC.

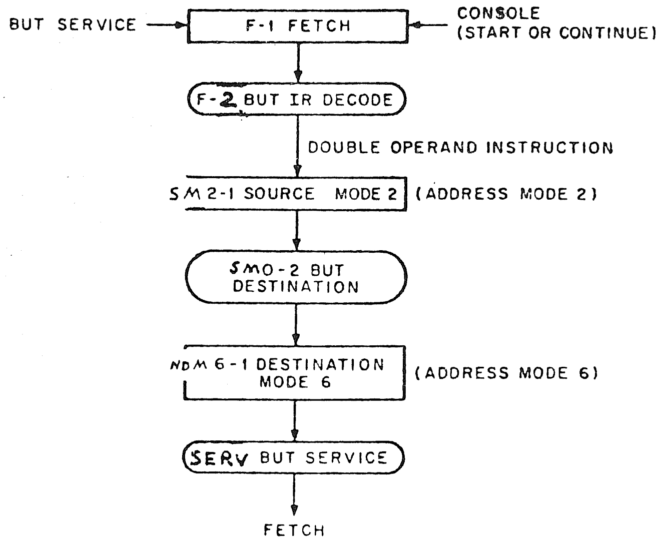


Figure 34 CMP # 15, CHAR (022767), Simplified Flow Diagram

LOCATION	NEXT LOCATION	MICROSTEP NAME	ACTION	COMMENT
102	123	F1	BA_PC,DATI, B_IR_UNIBUS DATA, J/F2	Drive the UNIBUS ADDRESS lines with the contents of the PC (R7) and initiate a DATI transfer. Load the data received from memory into both the B Register and the Instruction Register. Jump to the next microstep F2 (Loc. 123).
123	ØWIREORED F2 with offset of instruction decoded	PC_PC PLUS 2, BUT IR DECODE, J/SERV		Add two to the contents of the Program Counter Branch on Micro test to the instruction routine determined by the instruction decode logic.

Since the instruction is of the double operand group, the next step is to get the source data. Source mode 2 is autoincrement (Autoincrement implies one level of deferred addressing). When used with R7 (PC), it becomes an immediate mode.

LOCATION	NEXT LOCATION	MICROSTEP NAME	ACTION	COMMENT
62	114 WIREORED with BYTE status	SM2-1	BA_RS,DATI, ALBYTE, B_UNIBUS DATA, BUT BYTE,J/SM2Ø2	Place the contents of the source register specifed by IR (Ø8:Ø6) on the UNIBUS Address lines. The register will contain the location of the source data (1ØØ2) in this example. Initiate a UNIBUS DATI to actually get the data. ALBYT will allow an odd UNIBUS transfer, if the IR contains a byte instruction and the BA contains an odd address. Without the ALBYT, a UNIBUS transfer that addresses an odd BA results in a bus error. Input the data from memory into the B Register and microbranch to the following locations depending on whether a byte (odd or even) instruction is being performed. SM2-2 for not byte SMBE-1 for even bytes SMBO-1 for odd bytes.
114	1Ø4	SM2-2	RS_RS PLUS 2, J/SMØ-2	Add two to the contents of the source register. Microbranch to the next microstep SMØ-2 (1Ø4).
1Ø4	Ø WIREORD with dest mode	SMØ-2	R11_B, BUT DEST,J/SERV	Store source operand in Scratch Pad Register 11 and microbranch to the destination routine.

The microroutine starting in NDM6-1 will get the destination data and perform the operation indicated by the OP CODE of the instruction. Mode 6, when used with the PC, requires that the index contained in the word currently pointed to by the PC be added to the updated PC (address of the index word plus two) to get the location of the source data.

LOCATION	NEXT LOCATION	MICROSTEP NAME	ACTION	COMMENT
56	233	NDM6-1	BA_PC,DATI, B_UNIBUS DATA, J/NDM6-2	Perform a UNIBUS DATI transfer to obtain the index word and place it in the B Register. Microbranch to step NDM6-2.
233	235	NDM6-2	PC_PC PLUS 2, J/NDM6-3	Add two to the contents of the program counter and microbranch to NDM6-3.
235	23Ø	NDM6-3	B_B PLUS RD, J/NDM3-3	Add index word to contents of destination register specified by IR (2:Ø) to obtain address of destination operand.
23Ø	231	NDM3-3	R12_B, J/NDM3-4	TRANSFER address of destination operand to scratch pad register 12.
231	164	NDM3-4	BA_R12,DATI, ALBYT, b_unibus data, BUT BYTE,J/NDMØ-2	Place destination address (R12) on UNIBUS and perform UNIBUS DATI operation. ALBYT will allow an odd UNIBUS transfer if the IR contains a byte instruction. Input destination operand an store it in the B Register. Microbranch if byte instruction to one of the following. NDOB-1 if odd byte NDEB-1 if even byte NDMØ-2 if not byte.
164	Ø	NDMØ-2	B_R11 OP B, COND CODES, J/SERV	Operate (CMP) on source and destination operands and set condition codes according to result. Microbranch to SERVICE.
Ø	1Ø2	SERV	B_UNIBUS DATA, BUT SERVICE,J/F1	At the end of each instruction, various situations that attempt to intervene before the next instruction is fetched. Their prioritires are arbitrated as shown in Table 13. If no conditions with higher priority exist, the microprogram proceeds to the next FETCH (F1).

This completes the example of the microprogarm interpretation of CMP #5,CHAR. It may be useful to trace this or some other instruction through the detailed flow diagrams available in the KD11D print set.

6.3.2 Interrupts and Traps

Interrupts and traps are also accomplished by the microprogram. The follow is the microcode necessary for these routines.

LOCATION	NEXT LOCATION	MICROSTEP NAME	ACTION	COMMENT
0	102	SERV	B_UNIBUS DATA, BUT SERVICE, J/F1	;BRANCH ON SERVICE REQUEST ;LOAD VECTOR INTO BREG ;IF SERVICE REQUEST GO TO TRAP-1 ;IF NOT SERVICE REQUEST GO TO F1
103	20	TRAP-1:	R13_B,J/TRAP-2	;MOVE CONTENTS OF B REGISTER ;TO SP REGISTER 13
20	101	TRAP-2:	R6_R6 MINUS 2, ENABOVER, J/TRAP-3	;SUBTRACT TWO FROM STACK ;POINTER, PERMIT OVERFLOW
101	105	TRAP-3:	BA_R6,DATO, ENAB DBE, UNIBUS DATA_PSW, J/TRAP-4	;OUTPUT PROCESSOR STATUS TO ;STACK, ENABLE DOUBLE ;BUS ERROS
105	110	TRAP-4:	R6_R6 MINUS 2, ENABOVER, J/TRAP-5	;SUBTRACT TWO FROM STACK POINTER ;ALLOW OVERFLOW
110	111	TRAP-5:	B_PC,J/TRAP-6	;MOVE CONTENTS OF PC TO B REGISTER
111	112	TRAP-6:	BA_R6,DATO, UNIBUS DATA_B, J/TRAP-7	;OUTPUT PC TO STACK
112	113	TRAP-7:	NOP,J/TRAP-8	;MOCK MIRCO-STEP
113	115	TRAP-8:	BA_R13,DATI, B_UNIBUS DATA, J/TRAP-9	;INPUT NEW PC FROM MEMORY ADDRESS SPECIFIED BY SP REGISTER
115	120	TRAP-9:	R13_R13 PLUS 2, J/TRAP-10	;ADD TWO TO SP REGISTER 13
120	121	TRAP-10:	PC_B,J/TRAP-11	;LOAD NEW PC
121	122	TRAP-11:	BA_R13,DATI, B_UNIBUS DATA, J/TRAP-12	;INPUT NEW PROCESOR STATUS INTO REGISTER ;FROM LOCATION SPECIFIED BY ;SP REGISTER 13.

122	0	TRAP-12:	PSW_B,J/SERV	;LOAD NEW PROCESSOR STATUS ;INTO PSW REGISTER

6.3.3 Restarts From Power Failure

Upon restarting the KD11D, the processor begins running the microcode routine at MPC location one. This routine allows the processor to obtain its PC (program counter) and PSW (Processor Status Word) from memory and then begin running the program specified. This Restart routine is as follows.

LOCATION	NEXT LOCATION	MICROSTEP NAME	ACTION	COMMENT
1	362	REST-1:	B_PC,J/REST-2	;PROGRAM COUNTER TO B REGISTER
362	363	REST-2:	R5_B,J/REST-3	;MOVE B REGISTER TO REGISTER 5
363	364	REST-3:	R13_0,J/REST-4	;ZERO SP REGISTER
364	365	REST-4:	R13_R13 PLUS 2, J/REST-5	;PERFORM NEXT 5 STEPS TO
365	366	REST-5:	R13_R13 PLUS R13, J/REST-6	;OPTAIN 24 AS THE CONTENTS
366	367	REST-6:	R13_R13 PLUS 1, J/REST-7	;OF SP REGISTER 13
367	370	REST-7:	R13_R13 PLUS R13, J/REST-8	
370	113	REST-8:	R13_R13 PLUS R13, J/TRAP-8	
113	115	TRAP-8:	BA_R13,DATI, B_UNIBUS DATA, J/TRAP-9	;INPUT NEW PC FROM MEMORY ;ADDRESS SPECIFIED BY SP REGISTER
115	120	TRAP-9:	R13_R13 PLUS 2, J/TRAP-10	;ADD TWO TO SP REGISTER 13
120	121	TRAP-10:	PC_B,J/TRAP-11	;LOAD NEW PC
121		TRAP-11:	BA_R13,DATI, B_UNIBUS DATA, J/TRAP-12	;INPUT NEW PROCESSOR STATUS INTO REGISTER B ;FROM LOCATION SPECIFIED BY ;SP REGISTER 13.
122	0	TRAP-12:	PSWB,J/SERV	;LOAD NEW PROCESSOR STATUS ;INTO PSW REGISTER